aws

Guida per l'utente

Amazon Neptune

Amazon Neptune: Guida per l'utente

Table of Contents

Che cos'è Amazon Neptune?

Amazon Neptune è un servizio di database a grafo gestito rapido e affidabile che rende più semplice la creazione e l'esecuzione di applicazioni che funzionano con set di dati altamente connessi. Il cuore di Neptune è un motore di database a grafo ad alte prestazioni appositamente progettato. Questo motore è ottimizzato per archiviare miliardi di relazioni ed eseguire query sul database a grafo con una latenza di millisecondi. Neptune supporta i linguaggi di query per i grafi di proprietà più diffusi come Apache TinkerPop Gremlin, openCypher di Neo4j e SPARQL, il linguaggio di query RDF del W3C. Ti consente di creare query che interrogano in modo efficiente set di dati altamente connessi. Neptune abilita i casi d'uso dei grafi come motori di raccomandazioni, rilevamento di attività fraudolente, grafi della conoscenza, scoperte di farmaci e sicurezza delle reti.

Il database Neptune è altamente disponibile, con repliche di lettura, ripristino point-in-time (PITR), backup continuo in Amazon S3 e replica tra zone di disponibilità. Neptune offre funzionalità di sicurezza dei dati e supporta la crittografia dei dati inattivi e in transito. Neptune è una soluzione completamente gestita, pertanto non dovrai più preoccuparti delle attività di gestione del database come provisioning dell'hardware, applicazione di patch del software, installazione, configurazione o backup.

Analisi Neptune è un motore di database di analisi che integra il database Neptune e che può analizzare rapidamente grandi quantità di dati grafici in memoria per ottenere informazioni e scoprire tendenze. Analisi Neptune è una soluzione per analizzare rapidamente i database a grafo esistenti o i set di dati grafici archiviati in un data lake. Utilizza i più diffusi algoritmi di analisi grafica e query analitiche a bassa latenza.

Per ulteriori informazioni su Amazon Neptune, ti consigliamo di iniziare con le seguenti sezioni:

- Nozioni di base su Amazon Neptune
- Panoramica delle funzionalità di Amazon Neptune

Se non hai familiarità con i grafi o non sei ancora pronto a investire in un ambiente di produzione Neptune completo, consulta il nostro argomento Nozioni di base per scoprire come utilizzare i notebook Jupyter Neptune per l'apprendimento e lo sviluppo senza l'addebito di costi.

Prima di iniziare a progettare un database, ti consigliamo inoltre di consultare il repository GitHub AWSAWS Reference Architectures for Using Graph Databases, in cui troverai informazioni per le

scelte relative ai modelli di dati a grafo e ai linguaggi di query e dove potrai esplorare esempi di architetture di implementazione di riferimento.

Componenti dei servizi chiave

- Istanza database primaria – Supporta operazioni di lettura e scrittura ed esegue tutte le modifiche ai dati del volume del cluster. Ogni cluster database Neptune include un'istanza database primaria responsabile della scrittura, ad esempio il caricamento o la modifica, del contenuto del database a grafo.
- Replica Neptune: si connette allo stesso volume di archiviazione dell'istanza database primaria e supporta solo operazioni di lettura. Ogni cluster di database di Neptune può avere fino a 15 repliche di Neptune, oltre all'istanza database principale. In questo modo si garantisce disponibilità elevata posizionando le repliche di Neptune in zone di disponibilità separate e distribuendo il carico dei client di lettura.
- Volume del cluster: i dati Neptune vengono archiviati nel volume del cluster, concepito per garantire affidabilità e disponibilità elevata. Un volume del cluster è costituito da copie di dati distribuite su più zone di disponibilità in una singola regione AWS. Poiché vengono replicati automaticamente nelle zone di disponibilità, i dati risultano estremamente durevoli e poco soggetti ad andare perduti.

Supporta le API Open Graph

Amazon Neptune supporta le API Open Graph sia per i grafi di proprietà (Gremlin e openCypher) che per i grafi RDF (SPARQL). Fornisce prestazioni elevate per entrambi i modelli di grafo e i loro linguaggi di query. Puoi scegliere il modello Property Graph (PG) e accedere allo stesso grafo sia con il linguaggio di query openCypher che con il linguaggio di query Gremlin. Se usi il modello Resource Description Framework (RDF) standard del W3C, puoi accedere al grafo con il linguaggio di query SPARQL standard.

Massima sicurezza

Neptune offre più livelli di sicurezza per il database. Le funzionalità di sicurezza includono l'isolamento della rete tramite Amazon VPC e la crittografia dei dati inattivi tramite chiavi create e controllate tramite AWS Key Management Service (AWS KMS). In un'istanza Neptune crittografata, i dati archiviati nello spazio di archiviazione sottostante sono crittografati, così come snapshot, repliche e backup automatici nello stesso cluster.

Servizio interamente gestito

Con Amazon Neptune non dovrai più preoccuparti delle attività di gestione del database come provisioning dell'hardware, applicazione di patch software, installazione, configurazione, monitoraggio o backup.

Puoi utilizzare Neptune per creare applicazioni a grafo sofisticate e interattive capaci di interrogare miliardi di relazioni in millisecondi. Le query SQL per i dati altamente connessi sono complesse e difficili da mettere a punto per le prestazioni. Neptune ti permette di utilizzare i linguaggi di query per i grafi più diffusi come Gremlin, openCypher e SPARQL per eseguire potenti query facili da scrivere e che garantiscono prestazioni ottimali sui dati connessi. Questa funzione consente di ridurre significativamente la complessità del codice e di creare rapidamente applicazioni in grado di elaborare le relazioni.

Neptune è progettato per offrire una disponibilità superiore al 99,99%. Aumenta le prestazioni e la disponibilità del database grazie alla perfetta integrazione del motore di database con un livello di archiviazione virtualizzato supportato da SSD, creato per i carichi di lavoro di database. L'archiviazione Neptune offre tolleranza agli errori e riparazione automatica. Gli errori dei dischi vengono riparati in background senza perdita di disponibilità del database. Neptune rileva automaticamente gli arresti anomali del database e si riavvia senza la necessità di un ripristino da arresto anomalo o di ricostruire la cache del database. Se l'errore riguarda l'intera istanza, Neptune eseguirà automaticamente il failover su una delle 15 repliche di lettura.

Modifiche e aggiornamenti ad Amazon Neptune

Nella tabella seguente sono descritte le modifiche importanti apportate ad Amazon Neptune.

Modifica	Descrizione	Data
Versione del motore 1.2.1.1	A partire dall'11 marzo 2024, la versione del motore 1.2.1.1 viene generalmente utilizzata. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, vedere Neptune Engine Release 1.2.1.1.	11 marzo 2024
Versione del motore 1.3.1.0	A partire dal 06/03/2020, la versione del motore 1.3.1.0 viene generalmente utilizzata. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, vedere Neptune Engine Release 1.3.1.0.	6 marzo 2024
Aggiornamento delle autorizzazioni delle policy AWS gestite	Le politiche `NeptuneReadOnlyAcess` e le politiche `NeptuneFullAccess` gestite ora includono `Sid` (ID della dichiarazione) come	22 gennaio 2024

	identificatore nell'informativa sulla politica.	
Neptune offre ora archiviazione ottimizzata per l'I/O	Con l'archiviazione ottimizzata per l'I/O, si paga solo per l'archiviazione e le istanze utilizzate. I costi di archiviazione sono più elevati rispetto all'archiviazione standard, ma non è previsto alcun pagamento per l'I/O utilizzato.	13 dicembre 2023
Modifiche alle policy gestite da IAM per Neptune	La policy gestita da NeptuneConsoleFullAccessIAM è stata aggiornata per concedere le autorizzazioni necessarie per interagire con i grafici di Neptune Analytics, è stata aggiunta una NeptuneGraphReadOnlyAccessnuova policy per fornire l'accesso in sola lettura alle risorse grafiche di Neptune Analytics ed è stata aggiunta una nuova `AWSServiceRoleForNeptuneGraphPolicy` policy per consentire ai grafici di Neptune Analytics di pubblicare metriche e log operativi e di utilizzo. CloudWatch	29 novembre 2023

Versione del motore 1.3.0.0	A partire dal 15 novembre 2023, la versione del motore 1.3.0.0 viene implementata a livello generale. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Neptune Engine Release 1.3.0.0.	15 novembre 2023
Neptune è stato lanciato nella regione Israele (Tel Aviv)	Amazon Neptune è ora disponibile nella regione Israele (Tel Aviv) (`il-central-1`).	13 novembre 2023
Post del blog sull'implementazione time-to-live nei grafici delle proprietà di Neptune	Consulta Implement Time to Live in Amazon Neptune, Part 1: Property Graph di Melissa Kwok, Mike Havey e Kevin Phillips.	27 ottobre 2023
Versione del motore 1.2.1.0.R7	A partire dal 6 ottobre 2023, la versione del motore 1.2.1.0.R7 viene implementata a livello generale. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.1.0.R7.	6 ottobre 2023

Versione del motore 1.2.0.0.R4	A partire dal 29 settembre 2023, viene implementata a livello generale la versione del motore 1.2.0.0.R4. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.0.0.R4.	29 settembre 2023
Versione del motore 1.2.0.1.R3	A partire dal 27 settembre 2023, viene implementata a livello generale la versione del motore 1.2.0.1.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.0.1.R3.	27 settembre 2023

Versione del motore 1.2.1.0.R6	A partire dal 12 settembre 2023, viene implementata a livello generale la versione del motore 1.2.1.0.R6. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.1.0.R6.	12 settembre 2023
Versione del motore 1.2.0.2.R6	A partire dal 12 settembre 2023, viene implementata a livello generale la versione del motore 1.2.0.2.R6. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.0.2.R6.	12 settembre 2023
Post del blog sull'utilizzo di una strategia di implement azione blu/verde per gli aggiornamenti del motore Neptune	Consulta Improve availability of Amazon Neptune during engine upgrade using blue/green deployment di Ankit Gupta e Abhishek Mishra.	11 settembre 2023

Versione del motore 1.2.1.0.R5	A partire dal 2 settembre 2023, viene implementata a livello generale la versione del motore 1.2.1.0.R5. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.1.0.R5.	2 settembre 2023
Versione del motore 1.2.0.2.R5	A partire dal 16 agosto 2023, viene implementata a livello generale la versione del motore 1.2.0.2.R5. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.0.2.R5.	16 agosto 2023

Versione del motore 1.2.1.0.R4	A partire dal 10 agosto 2023, viene implementata a livello generale la versione del motore 1.2.1.0.R4. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.1.0.R4.	10 agosto 2023
Post del blog sul rilascio 1.2.1.0 del motore Neptune	Consulta Exploring the feature packed 1.2.1.0 release for Amazon Neptune di Joy Wang, Kevin Phillips, Andrea Nassisi e Navtanay Sinha.	4 agosto 2023
Post del blog sulla creazione di una soluzione di database multimodale con Neptune	Consulta Design a use case-driven, highly scalable multimodel database solution using Amazon Neptune di Mike Havey.	18 luglio 2023
Post del blog sui casi d'uso e sulle best practice di Neptune Serverless	Consulta Use cases and best practices to optimize cost and performance with Amazon Neptune Serverless di Kevin Phillips e Ankit Gupta.	28 giugno 2023

Versione del motore 1.2.1.0.R3	A partire dal 13 giugno 2023, viene implementata a livello generale la versione del motore 1.2.1.0.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.1.0.R3.	13 giugno 2023
Post del blog sulla generazione di suggerimenti per il tempo libero in tempo reale	Consulta Generate suggestions for leisure activities in real time with Amazon Neptune di Michael Meidlinger e Nils Müller.	6 giugno 2023
Post del blog sulla modellazione molecolare con Neptune e RDKit	Consulta Model molecular SMILES data with Amazon Neptune and RDKit di Graham Kutchek.	1 giugno 2023
Post del blog sull'uso della memorizzazione nella cache per accelerare le prestazioni di Neptune (parte 3)	Vedi Accelerare le prestazioni delle query grafiche con la memorizzazione nella cache in Amazon Neptune, parte 3: architetture di caching a livello di cluster Neptune con Amazon di Taylor Riggan, Abhishek Mishra, Melissa Kwok e Kelvin ElastiCache Lawrence.	26 maggio 2023

Post del blog sull'uso della memorizzazione nella cache per accelerare le prestazioni di Neptune (parte 2)	Consulta Accelerate graph query performance with caching in Amazon Neptune, Part 2: Additional Neptune caching features di Taylor Riggan, Abhishek Mishra, Melissa Kwok e Kelvin Lawrence.	26 maggio 2023
Post del blog sull'uso della memorizzazione nella cache per accelerare le prestazioni di Neptune (parte 1)	Consulta Accelerate graph query performance with caching in Amazon Neptune, Part 1: Queries and buffer pool caching di Taylor Riggan, Abhishek Mishra, Melissa Kwok e Kelvin Lawrence.	26 maggio 2023
Post del blog sull'analisi della catena di approvvigionamento con Neptune	Consulta Supply chain data analysis and visualization using Amazon Neptune and the Neptune workbench di Dhiraj Thakur e Rajdip Chaudhur.	10 maggio 2023
Versione del motore 1.2.0.2.R4	A partire dall'8 maggio 2023, viene implementata a livello generale la versione del motore 1.2.0.2.R4. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.0.2.R4.	8 maggio 2023

Neptune è stato lanciato in Medio Oriente (Emirati Arabi Uniti)	Amazon Neptune è ora disponibile in Medio Oriente (Emirati Arabi Uniti) (`me-central-1`).	2 maggio 2023
Versione del motore 1.2.1.0.R2	A partire dal 2 maggio 2023, viene implementata a livello generale la versione del motore 1.2.1.0.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.1.0.R2.	2 maggio 2023
Post del blog sulla creazione di un grafo della conoscenza su Neptune con analisi video basata sull'intelligenza artificiale utilizzando Media2Cloud	Consulta Build a knowledge graph on Amazon Neptune with AI-powered video analysis using Media2Cloud di Mike Havey.	2 maggio 2023
Post di blog su come è DevOcean stata creata una piattaforma di correzione delle vulnerabilità utilizzando Neptune	Scopri come ha DevOcean creato una piattaforma di gestione della correzione delle vulnerabilità per applicazioni native del cloud utilizzando Amazon Neptune di Gil Makmel e Charles Ivie.	25 aprile 2023

Post del blog su come Getir ha creato un sistema di rilevamento delle frodi utilizzando Neptune	Consulta How Getir build a comprehensive fraud detection system using Amazon Neptune and Amazon DynamoDB di Berkay Berkman, Mahmut Turan, Mutlu Polatcan, Umut Cemal Kıraç, Yağız Yanıkoğlu ed Esra Kayabali.	6 aprile 2023
Post del blog su come Wiz reinventa la sicurezza del cloud utilizzando Neptune	Consulta The World is a graph: How Wiz reimagines cloud security using a graph in Amazon Neptune di Ami Luttwak e Brad Bebee.	31 marzo 2023
Versione del motore 1.2.0.2.R3	A partire dal 27 marzo 2023, viene implementata a livello generale la versione del motore 1.2.0.2.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.0.2.R3.	27 marzo 2023
Post del blog su come CSC Generation promuove l'individuazione dei prodotti utilizzando Neptune	Consulta How CSC Generation powers product discovery with knowledge graphs using Amazon Neptune di Bobber Cheng, Ronit Rudra e Melissa Kwok.	21 marzo 2023

Versione del motore 1.2.1.0	A partire dall'8 marzo 2023, viene implementata a livello generale la versione del motore 1.2.1.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.1.0.	8 marzo 2023
Post di blog sull'esplorazione delle nuove funzionalità di TinkerPop 3.6.x in Neptune	Vedi Esplorazione delle nuove funzionalità di Apache TinkerPop 3.6.x in Amazon Neptune di Stephen Mallette.	8 marzo 2023
Post del blog sull'utilizzo del ragionamento semantico per dedurre nuovi fatti dal grafo RDF (Resource Description Framework)	Consulta Use semantic reasoning to infer new facts from your RDF graph by integrating RDFox with Amazon Neptune di Charles Ivie and Diana Marks.	20 febbraio 2023
Post del blog sull'analisi dei dati FHIR del settore sanitario con Neptune	Consulta Analyze healthcare FHIR data with Amazon Neptune di Alena Schmickl.	13 febbraio 2023
Post del blog sulla creazione di una soluzione di rilevamento delle frodi in tempo reale utilizzando Neptune ML	Consulta Build a real-time fraud detection solution using Amazon Neptune ML di Hua Shu e Soji Adeshina.	8 febbraio 2023

Versione del motore 1.1.1.0.R7	A partire dal 23 gennaio 2023, viene implementata a livello generale la versione del motore 1.1.1.0.R7. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.1.1.0.R7.	23 gennaio 2023
È stato rilasciato graph-explorer	Graph-explorer è uno strumento per applicazioni Web front-end open source per la visualizzazione dei dati dei grafi. Consulta https://github.com/aws/graph-explorer.	3 gennaio 2023
Rilascio di manutenzione della versione 1.1.0.0.R3	A partire dal 23 dicembre 2022, il rilascio di manutenzione della versione del motore 1.1.0.0.R3 è implementato a livello generale. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.1.0.0.R3.	23 dicembre 2022

L'ambiente di lavoro Neptune ora funziona su Amazon Linux 2 e 3. JupyterLab	I notebook Neptune Graph ora funzionano in un ambiente Amazon Linux 2 con 3. JupyterLab Vedi Migrazione dei notebook Neptune da Jupyter a 3 JupyterLab per informazioni su come passare a questo nuovo ambiente.	21 dicembre 2022
Post del blog su come abilitare le raccomandazioni e la ricerca utilizzando un grafo della conoscenza di IMDb (parte 3)	Consulta Power recommendations and search using an IMDb knowledge graph – Part 3 di Divya Bhargavi, Soji Adeshina, Gaurav Rele, Karan Sindwani, Vidya Sagar Ravipati e Matthew Rhodes.	20 dicembre 2022
Post del blog su come abilitare le raccomandazioni e la ricerca utilizzando un grafo della conoscenza di IMDb (parte 2)	Consulta Power recommendations and search using an IMDb knowledge graph – Part 2 di Matthew Rhodes, Soji Adeshina, Divya Bhargavi, Gaurav Rele, Karan Sindwani e Vidya Sagar Ravipati.	20 dicembre 2022
Post del blog su come abilitare le raccomandazioni e la ricerca utilizzando un grafo della conoscenza di IMDb (parte 1)	Consulta Power recommendations and search using an IMDb knowledge graph – Part 1 di Gaurav Rele, Soji Adeshina, Divya Bhargavi, Karan Sindwani, Vidya Sagar Ravipati e Matthew Rhodes.	20 dicembre 2022

Neptune Serverless è ora disponibile in nuove regioni AWS	A partire dal 16 dicembre 2022, Neptune Serverless è stato lanciato nelle nuove regioni AWS seguenti: Canada (Centrale), Europa (Stoccolma), Europa (Francoforte), Asia Pacifico (Singapore) e Asia Pacifico (Sydney). Scopri i Vincoli di Amazon Neptune Serverless per tutte le regioni in cui è disponibile Neptune Serverless.	16 dicembre 2022
Versione del motore 1.2.0.2.R2	A partire dal 15 dicembre 2022, viene implementata a livello generale la versione del motore 1.2.0.2.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.0.2.R2.	15 dicembre 2022

Versione del motore 1.2.0.0.R3	A partire dal 15 dicembre 2022, viene implementata a livello generale la versione del motore 1.2.0.0.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.0.0.R3.	15 dicembre 2022
Versione del motore 1.2.0.1.R2	A partire dal 13 dicembre 2022, viene implementata a livello generale la versione del motore 1.2.0.1.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.0.1.R2.	13 dicembre 2022
Post di blog sulla progettazione di un'architettura didattica per l'analisi dei big data con AWS	Consulta Designing an educational big data analysis architecture with AWS di Lavanya Sood.	13 dicembre 2022
Post del blog su come migliorare l'apprendimento con i database a grafo	Consulta How graph databases can enhance learning di Lavanya Sood.	8 dicembre 2022

Post sul blog sul caricamento di dati RDF in Neptune usando Glue AWS	Vedi Caricare dati RDF in Amazon AWS Neptune with Glue di Mike Havey e Fabrizio Napolitano.	23 novembre 2022
Versione del motore 1.2.0.2	A partire dal 20 novembre 2022, viene implementata a livello generale la versione del motore 1.2.0.2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.0.2.	20 novembre 2022
Versione del motore 1.2.0.1	A partire dal 26 ottobre 2022, viene implementata a livello generale la versione del motore 1.2.0.1. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.0.1.	26 ottobre 2022
Post del blog sul rilevamento delle frodi utilizzando Neptune	Consulta Empowering fraud detection at Delivery Hero with Amazon Neptune di Wilson Tang, Amr Elnaggar, Matias Pons, Mohammad Azzam, Saurabh Deshpande e Luis Rodrigues Soares.	26 ottobre 2022

Post del blog su Neptune Serverless	Consulta Introducing Amazon Neptune Serverless – A Fully Managed Graph Database that Adjusts Capacity for Your Workloads di Danilo Poccia.	26 ottobre 2022
Post del blog sulle importazioni RDF basate su eventi in Neptune utilizzando Lambda e SPARQL UPDATE LOAD	Scopri come NXP esegue importazioni RDF basate su eventi in Amazon Neptune utilizzando AWS Lambda e SPARQL UPDATE LOAD di John Walker, Onno Buijs, Charles Ivie e Javy de Koning.	20 ottobre 2022
Versione del motore 1.2.0.0.R2	A partire dal 14 ottobre 2022, viene implementata a livello generale la versione del motore 1.2.0.0.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.0.0.R2.	14 ottobre 2022
Post del blog sulla codifica di proprietà del testo multilingue in Neptune	Consulta Encode multi-lingual text properties in Amazon Neptune to train predictive models di Jiani Zhang.	14 ottobre 2022
Post del blog sui test automatici dell'accesso ai dati di Neptune	Vedi Test automatizzati dell'accesso ai dati di Amazon Neptune con TinkerPop Apache Gremlin di Greg Biegel.	28 settembre 2022

Versione del motore 1.1.1.0.R6	A partire dal 23 gennaio 2023, viene implementata a livello generale la versione del motore 1.1.1.0.R6. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.1.1.0.R6.	23 settembre 2022
Post del blog su come Informatica® utilizza Neptune	Consulta How Informatica® Cloud Data Governance and Catalog uses Amazon Neptune for knowledge graphs di Tiju Titus John, Deepak Ram e Farooq Ashraf.	20 settembre 2022
Post del blog sull'utilizzo di Neptune e Tom Sawyer Perspectives per rilevare frodi finanziarie	Consulta Uncover financial fraud with Amazon Neptune and Tom Sawyer Perspectives di Janet M. Six, Senior Product Manager presso Tom Sawyer Software.	30 agosto 2022
Post di blog sulla creazione di una soluzione di rilevamento delle frodi in tempo reale basata su GNN utilizzando SageMaker Neptune e DGL	Vedi Creare una soluzione di rilevamento delle frodi in tempo reale basata su GNN utilizzando Amazon SageMaker, Amazon Neptune e la Deep Graph Library di Jian Zhang, Haozhu Wang e Mengxin Zhu.	11 agosto 2022

Post del blog sull'utilizzo dei tag delle risorse per arrestare e avviare le risorse dell'ambiente Neptune	Consulta Automate the stopping and starting of Amazon Neptune environment resources using resource tags di Kevin Phillips.	1 agosto 2022
Post di blog su un artista che contribuisce ad Apache TinkerPop	Vedi Beyond Code: L'artista che contribuisce ad TinkerPop Apache di Stephen Mallette e Ketrina Thompson.	1 agosto 2022
Post del blog sul controllo granulare degli accessi per le azioni del piano dati di Neptune	Consulta Fine Grained Access Control for Amazon Neptune data plane actions di Abhishek Mishra e Ankit Gupta.	29 luglio 2022
Post del blog sui database globali di Neptune	Consulta Introducing Amazon Neptune Global Database di Navtanay Sinha.	27 luglio 2022
Versione del motore 1.2.0.0	A partire dal 21 luglio 2022, viene implementata a livello generale la versione del motore 1.2.0.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.2.0.0.	21 luglio 2022

Versione del motore 1.1.1.0.R5	A partire dal 21 luglio 2022, viene implementata a livello generale la versione del motore 1.1.1.0.R5. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.1.1.0.R5.	21 luglio 2022
Versione del motore 1.1.1.0.R4	A partire dal 23 giugno 2022, viene implementata a livello generale la versione del motore 1.1.1.0.R4. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.1.1.0.R4.	23 giugno 2022

Flussi di lavoro semplificati di analisi dei grafi e machine learning con l'integrazione Python	Ora puoi eseguire attività di analisi dei grafi e machine learning su dati dei grafi archiviati in Amazon Neptune utilizzando un'integrazione Python open source che semplifica i flussi di lavoro di data science e machine learning. Consulta la Documentazione di AWS Data Wrangler per Neptune.	7 luglio 2022
Versione del motore 1.1.1.0.R3	A partire dal 7 giugno 2022, viene implementata a livello generale la versione del motore 1.1.1.0.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.1.1.0.R3.	7 luglio 2022
Post del blog sul rilevamento di fake news	Consulta Detect social media fake news using graph machine learning with Amazon Neptune ML di Hasan Shojaei e Sarita Joshi.	19 maggio 2022
Post del blog sull'utilizzo di SQL Server Integration Services (SSIS) con Neptune	Consulta Discover new insights from your data using SQL Server Integration Services (SSIS) and Amazon Neptune di Mesgana Gormley e Melissa Kwok.	18 maggio 2022

Rilascio di manutenzione della versione 1.1.1.0.R2	A partire dal 16 maggio 2022, il rilascio di manutenzione della versione del motore 1.1.1.0.R2 è implementato a livello generale. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.1.1.0.R2.	16 maggio 2022
Rilascio di manutenzione della versione 1.1.0.0.R2	A partire dal 16 maggio 2022, il rilascio di manutenzione della versione del motore 1.1.0.0.R2 è implementato a livello generale. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.1.0.0.R2.	16 maggio 2022

Rilascio di manutenzione della versione 1.0.5.1.R4	A partire dal 16 maggio 2022, il rilascio di manutenzione della versione del motore 1.0.5.1.R4 è implementato a livello generale. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.5.1.R4.	16 maggio 2022
Rilascio di manutenzione della versione 1.0.5.0.R5	A partire dal 16 maggio 2022, il rilascio di manutenzione della versione del motore 1.0.5.0.R5 è implementato a livello generale. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.5.0.R5.	16 maggio 2022
Post del blog sulla disponibilità generale di openCypher in Neptune	Consulta Announcing the General Availability of openCypher support for Amazon Neptune di Navtanay Sinha e Dave Bechberger.	22 aprile 2022

Versione del motore 1.1.1.0	A partire dal 19 aprile 2022, viene implementata a livello generale la versione del motore 1.1.1.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.1.1.0.	19 aprile 2022
Post del blog sulla derivazione dei dati	Vedi Costruisci il data lineage per i data lake usando AWS Glue, Amazon Neptune e Spline, di Khoa Nguyen, Krithivasan Balasubramaniyan e Rahul Shaurya.	1 aprile 2022
Gli aggiornamenti al rilascio 1.1.0.0 del motore sono nuovamente abilitati	A partire dal 21 febbraio 2022 gli aggiornamenti al rilascio del motore 1.1.0.0 sono stati temporaneamente disabilitati. Ora sono nuovamente abilitati.	22 marzo 2022
Post del blog sull'affidabilità delle utilità di progettazione	Consulta Using cloud-based, data-informed, power system models to engineer utility reliability di Abhineet Parchure.	22 marzo 2022

Neptune è stato lanciato in Africa (Città del Capo)	Amazon Neptune è ora disponibile in Africa (Città del Capo) (`af-south-1`). Tuttavia, il supporto per notebook in Neptune Workbench è temporane amente disabilitato nella console Neptune in tale regione.	24 febbraio 2022
Post del blog sui grafi basati su modelli che utilizzano OWL	Consulta Model-driven graphs using OWL in Amazon Neptune di Mike Havey.	23 febbraio 2022
Post del blog sull'esplorazione dei grafi di conoscenza semantica con Rhizomer	Consulta Explore the semantic knowledge graphs without SPARQL using Amazon Neptune with Rhizomer di Roberto García.	22 febbraio 2022
Post del blog sulla rappresen tazione grafica della rete di distribuzione	Vedi AWS Graphing the utility grid on, di Bobby Wilson e Joseph Beer.	18 febbraio 2022
Nuove opzioni di codifica delle funzionalità del testo di Neptune ML	Neptune ora FastText supporta la codifica di testo Sentence BERT per l'addestr amento. Scopri FastTextle funzionalità di Neptune ML e le funzionalità di Sentence BERT in Neptune ML.	15 febbraio 2022
Post di blog sull'utilizzo di query geospaziali con Neptune OpenSearch	Vedi Combinare Amazon Neptune e OpenSearch Amazon Service per le query geospaziali, di Ross Gabay e Abhilash Vinod.	1 febbraio 2022

Post del blog sull'individuazione di reati finanziari con Amazon EKS e Neptune	Consulta Financial Crime Discovery using Amazon EKS and Graph Databases di Severin Gassauer-Fleissner e Zahi Ben Shabat.	1 febbraio 2022
Le dimensioni di un volume del cluster Neptune possono aumentare fino a 128 tebibyte (TiB)	In tutte le regioni supportate, ad eccezione della Cina GovCloud, il limite di dimensione di un volume del cluster Neptune è ora aumentato da 64 TiB a 128 TiB. Questo vale per tutti i rilasci del motore a partire dal rilascio 1.0.2.2. Consulta la pagina relativa all'archiviazione di Amazon Neptune.	1 febbraio 2022
La ricerca full-text di Neptune ora si integra con tutte le versioni di. OpenSearch	Consulta Ricerca nel testo completo in Amazon Neptune con OpenSearch Amazon Service.	28 gennaio 2022
Post del blog sull'utilizzo dei container Docker per la distribuzione di notebook per grafi	Vedi Usare i contenitori Docker per distribuire Graph Notebooks su, di Ganesh Sawhney e Qiang AWS Zhang.	22 gennaio 2022

Versione del motore 1.0.5.1.R3	A partire dal 13 gennaio 2022, viene implementata a livello generale la versione del motore 1.0.5.1.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.5.1.R3.	13 gennaio 2022
Post del blog su un sistema di raccomandazione basato su grafo che utilizza Neptune ML	Consulta Graph-based recommendation system with Neptune ML: An illustration on social network link predictio n challenges di Yanwei Cui e Will Badr.	12 gennaio 2022
Post del blog sul dimension amento automatico di Neptune	Consulta Auto scale your Amazon Neptune database to meet workload demands di Navtanay Sinha e Sudhanshu Gupta.	29 novembre 2021
Post del blog sull'analisi e le visualizzazioni interattive dei dati dei grafi	Vedi Creare analisi e visualizz azioni interattive di dati grafici utilizzando Amazon Neptune, Amazon Athena Federated Query e Amazon, di Sandeep Veldi QuickSight e Abhishek Mishra.	24 novembre 2021

Post del blog sul rilevamento delle frodi di identità mediante deep learning basato su grafo	Consulta How Careem is detecting identity fraud using graph-based deep learning and Amazon Neptune di Kevin O'Brien, Kamran Habib e Will Badr.	23 novembre 2021
Versione del motore 1.1.0.0	A partire dal 19 novembre 2021, viene implementata a livello generale la versione del motore 1.1.0.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.1.0.0.	19 novembre 2021
Post del blog sulla centralizzazione della protezione e della conformità dei dati	Vedi Centralizzazione della protezione e della conformità dei dati in Amazon Neptune with AWS Backup, di Brian O'Keefe.	8 novembre 2021
Post del blog sulla lotta alle frodi e ai pagamenti impropri	Consulta Fighting fraud and improper payments in real-time at the scale of federal expenditures di Vladi Royzman e Spencer Smith.	2 novembre 2021
Post del blog sulla prevenzione delle iscrizioni di account falsi	Consulta Prevent fake account sign-ups in real time with AI using Amazon Fraud Detector di Anjan Biswas.	29 ottobre 2021

Versione del motore 1.0.5.1.R2	A partire dal 26 ottobre 2021, viene implementata a livello generale la versione del motore 1.0.5.1.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.5.1.R2.	26 ottobre 2021
Post sul blog sulla previsione a HawkEye 360 gradi del rischio delle navi utilizzando la Deep Graph Library	See HawkEye 360 prevede il rischio delle navi utilizzando Deep Graph Library e Amazon Neptune di Tim Pavlick, Ian Avilez, Dan Ford e Gaurav Rele.	15 ottobre 2021
Versione del motore 1.0.5.1	A partire dal 1° ottobre 2021, viene implementata a livello generale la versione del motore 1.0.5.1. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.5.1.	1° ottobre 2021

Post sul blog che spiega perché piace agli sviluppatori TinkerPop	Scopri Perché agli sviluppatori piace Apache TinkerPop, un framework open source per il calcolo grafico, di Brad Bebee, Kelvin Lawrence e Stephen Mallette.	27 settembre 2021
Versione del motore 1.0.5.0.R3	A partire dal 15 settembre 2021, viene implementata a livello generale la versione del motore 1.0.5.0.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.5.0.R3.	15 settembre 2021
Post del blog sulla creazione di una soluzione di rilevamento dei dati con Amundsen e Neptune	Consulta Building a data discovery solution with Amundsen and Amazon Neptune di Peter Hanssens e Don Simpson.	8 settembre 2021
Neptune ha aggiornato lo strumento per il polling dei flussi per supportare query di ricerca full-text non di tipo stringa	In questo rilascio sono inclusi molti miglioramenti alla ricerca full-text, incluso il supporto per l'indicizzazione dei valori delle proprietà non di tipo stringa. Vedi OpenSearch Indicizzazione senza stringhe in Amazon Neptune.	23 agosto 2021

Versione del motore 1.0.5.0.R2	A partire dal 16 agosto 2021, viene implementata a livello generale la versione del motore 1.0.5.0.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.5.0.R2.	16 agosto 2021
Versione del motore 1.0.4.2.R5	A partire dal 16 agosto 2021, viene implementata a livello generale la versione del motore 1.0.4.2.R5. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.4.2.R5.	16 agosto 2021
Post del blog sul supporto per il protocollo GSP (Graph Store Protocol) in Neptune	Consulta Introducing Graph Store Protocol support for Amazon Neptune di Chris Smith.	2 agosto 2021
Post del blog sulle nuove funzionalità di Neptune ML	Consulta Discover more insights in your graphs with new features from Amazon Neptune ML di Soji Adeshina.	30 luglio 2021

Post del blog su come ottenere previsioni più velocemente con Neptune ML	Consulta Get predictions for evolving graph data faster with Amazon Neptune ML di Soji Adeshina.	30 luglio 2021
Post del blog sul machine learning basato su grafo più facile e veloce con Neptune ML	Consulta Easier and faster graph machine learning with Amazon Neptune ML di Soji Adeshina.	30 luglio 2021
Post del blog sul supporto di openCypher per Neptune	Consulta Announcing openCypher for Amazon Neptune: Building better graph applications with openCypher and Gremlin together, di Brad Bebee.	29 luglio 2021
Versione del motore 1.0.5.0	A partire dal 27 luglio 2021, viene implementata a livello generale la versione del motore 1.0.5.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.5.0.	27 luglio 2021

Versione del motore 1.0.4.2.R4	A partire dal 23 luglio 2021, viene implementata a livello generale la versione del motore 1.0.4.2.R4. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.4.2.R4.	23 luglio 2021
Neptune è stato lanciato in Cina (Pechino)	Amazon Neptune è ora disponibile in Cina (Pechino) (`cn-north-1`).	21 luglio 2021
Versione del motore 1.0.4.2.R3	A partire dal 28 giugno 2021, viene implementata a livello generale la versione del motore 1.0.4.2.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.4.2.R3.	28 giugno 2021
Post del blog su come Dream11 ha dimensionato il proprio social network utilizzando Neptune	Scopri come Dream11, la piattaforma di sport fantasy più grande al mondo, amplia il proprio social network con Amazon ElastiCache Neptune e Amazon.	25 giugno 2021

Post di blog sulla trasformazione dei dati in conoscenza con PoolParty Neptune e Semantic Suite	Vedi Trasforma i dati in conoscenza con PoolParty Semantic Suite e Amazon Neptune, di Ioanna Lytra e Albin Ahmeti.	16 giugno 2021
Post di blog sull'uso di Neptune per esplorare la knowledge base UniProt	Vedi Esplorazione della knowledge base sulle UniProt proteine con AWS Open Data e Amazon Neptune, di Eric Greene, Rafa Xu e Yuan Shi.	10 giugno 2021
Post del blog sull'utilizzo di Neptune per l'analisi del rischio basata sui dati	Vedi le note sul campo: Analisi del rischio basata sui dati con Amazon Neptune e OpenSearch Amazon Service, di Adriaan de Jonge e Rohit Satyanarayana.	10 giugno 2021
Versione del motore 1.0.4.2.R2	A partire dal 1° giugno 2021, viene implementata a livello generale la versione del motore 1.0.4.2.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.4.2.R2.	1 giugno 2021
Post di blog sull'uso di Neptune per visualizzare la tua infrastruttura AWS	Vedi Visualizza la tua AWS infrastruttura con Amazon AWS Neptune e Config, di Rohan Raizada e Amey Dhavle.	25 maggio 2021

Post del blog sulla configurazione per l'utilizzo di Data Lens con Neptune	Vedi Configurare AWS i servizi per creare un knowledge graph in Amazon Neptune utilizzando Data Lens, di Russell Waterson.	5 maggio 2021
Post del blog sulla creazione di un grafo della conoscenza in Neptune utilizzando Data Lens	Consulta Build a knowledge graph in Amazon Neptune using Data Lens di Russell Waterson.	5 maggio 2021
Le versioni del motore 1.0.1.0, 1.0.1.1 e 1.0.1.2 sono diventate obsolete	A partire da ora, non verrà creata alcuna nuova istanza database utilizzando queste versioni del motore o patch a esse correlate.	26 Aprile 2021
Traduzione in inglese di un case study sull'utilizzo di Neptune da parte del Ministry of Economy, Trade and Industry del Giappone	Consulta Japan's Ministry of Economy, Trade and Industry Powers gBizINFO Corporate Information Search Database with AWS.	31 marzo 2021
Post del blog sull'utilizzo di Neptune con Amazon Comprehend e Amazon Lex	Consulta Supercharge your knowledge graph using Amazon Neptune, Amazon Comprehend, and Amazon Lex di Dave Bechberger.	31 marzo 2021
Post del blog sull'utilizzo delle funzioni Lambda con Neptune	Vedi Usare le funzioni AWS Lambda con Amazon Neptune, di Ian Robinson.	26 marzo 2021

Versione del motore 1.0.4.1.R1.1	A partire dal 22 marzo 2021, viene implementata a livello generale la versione del motore 1.0.4.1.R1.1. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.4.1.R1.	22 marzo 2021
Versione del motore 1.0.4.1.R2.1	A partire dall'11 marzo 2021, viene implementata a livello generale la versione del motore 1.0.4.1.R2.1. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.4.1.R2.1.	11 marzo 2021
Post del blog sull'utilizzo dello strumento Graph Notebook open source di Neptune per la visualizzazione di grafi	Consulta Getting started with open source graph notebook for graph visualization di Joy Wang, Ora Lassila e Stephen Mallette.	10 marzo 2021
Tutorial sull'integrazione di Neptune con il motore di metadati e rilevamento dei dati Amundsen	Consulta How to use Amundsen with Amazon Neptune di Andrew Ciambrone .	2 marzo 2021

Versione del motore 1.0.4.1.R2	A partire dal 24 febbraio 2021, viene implementata a livello generale la versione del motore 1.0.4.1.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.4.1.R2.	24 febbraio 2021
Versione del motore 1.0.4.0.R2	A partire dal 24 febbraio 2021, viene implementata a livello generale la versione del motore 1.0.4.0.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.4.0.R2.	24 febbraio 2021

Versione del motore 1.0.3.0.R3	A partire dal 19 febbraio 2021, viene implementata a livello generale la versione del motore 1.0.3.0.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.3.0.R3.	19 febbraio 2021
Versione del motore 1.0.2.2.R6	A partire dal 19 febbraio 2021, viene implementata a livello generale la versione del motore 1.0.2.2.R6. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.2.2.R6.	19 febbraio 2021
Post del blog sulla creazione di un grafo della conoscenza utilizzando gli eventi di Amazon Comprehend	Consulta Building a knowledge graph in Amazon Neptune using Amazon Comprehend Events di Brian O'Keefe, Graham Horwood e Navtanay Sinha.	19 gennaio 2021

Post del blog sull'abilitazione di app per i dati dei grafi con poco codice	Consulta Enabling low code graph data apps with Amazon Neptune and Graphistry di Leo Meyerovich, Dave Bechberger e Taylor Riggan.	18 gennaio 2021
È stata aggiunta la documentazione relativa ai notebook per iniziare a utilizzare i dati dei grafi.	È stata aggiunta una sezione che si integra con Neptune Workbench e che consente di iniziare a creare i dati dei grafi e sviluppare applicazioni a grafo senza dover creare un cluster Neptune finché non si è pronti.	15 gennaio 2021
Post del blog sulla reimpostazione dei dati dei grafi di Neptune in pochi secondi	Consulta Resetting your graph data in Amazon Neptune in seconds di Niraj Jetly e Navtanay Sinha.	17 dicembre 2020
Post sul blog su come Novartis AG utilizza SageMaker Neptune con BERT	Vedi Novartis AG utilizza Amazon SageMaker e Amazon Neptune per creare e arricchire un knowledge graph utilizzando BERT, di Othmane Hamzaoui, Fatema Alkhanaizi e Viktor Malesevic.	14 dicembre 2020
Post del blog sulla creazione di un grafo della conoscenza con reti di argomenti	Consulta Building a knowledge graph with topic networks in Amazon Neptune di Edward Brown, Head of AI Projects, Eduardo Piairo, Architect, Marcia Oliveira, Lead Data Scientist e Jack Hampson, CEO presso Deeper Insights.	14 dicembre 2020

Versione del motore 1.0.4.1	A partire dall'08 dicembre 2020, viene implementata a livello generale la versione del motore 1.0.4.1. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.4.1.	8 dicembre 2020
Post del blog su come iniziare a utilizzare Neptune ML	Consulta How to get started with Neptune ML di George Karypis, Dave Bechberger e Karthik Bharathy.	8 dicembre 2020
Neptune ora dispone di un'API di reimpostazione rapida	Utilizzando l'API di reimpostazione rapida, è possibile eliminare in modo facile e veloce tutti i dati in un cluster database. Consulta API di reimpostazione rapida.	4 dicembre 2020
Post del blog sulla creazione di un grafo della conoscenza per la ricerca biologica presso Pendulum	Consulta Building a biological knowledge graph at Pendulum using Amazon Neptune di Connor Skennerton.	26 novembre 2020
Post sul blog sulle nuove funzionalità di TinkerPop Neptune della versione 3.4.8	Vedi Esplorazione delle nuove funzionalità di Apache TinkerPop 3.4.8 in Amazon Neptune, di Stephen Mallette.	18 novembre 2020

Post del blog sull'utilizzo del servizio di ricerca Amazon Kendra con Neptune	Consulta Incorporating your enterprise knowledge graph into Amazon Kendra di Yazdan Shirvany, Mohit Mehta e Dipto Chakravarty.	17 novembre 2020
Notifiche di eventi ora disponibili	Neptune ora supporta le notifiche di eventi che è possibile utilizzare per monitorare più facilmente i cluster database. Consulta Utilizzo della notifica di eventi Neptune.	29 ottobre 2020
Endpoint personalizzati ora disponibili	Neptune ora supporta endpoint personalizzati per un maggiore controllo della connessione alle istanze di database. Consulta Connessione agli endpoint Amazon Neptune.	29 ottobre 2020
Post di blog sull'utilizzo del AWS Database Migration Service (DMS) per popolare il grafico di Neptune	Vedi Compilazione del grafico in Amazon Neptune da un database relazionale utilizzando Database AWS Migration Service (DMS) — Parte 4: Mettere tutto insieme, di Chris Smith.	22 ottobre 2020
Post di blog sull'utilizzo del AWS Database Migration Service (DMS) per popolare il grafico di Neptune	Vedi Compilazione del grafico in Amazon Neptune da un database relazionale utilizzando Database AWS Migration Service (DMS) — Parte 3: Progettazione del modello RDF, di Chris Smith.	22 ottobre 2020

Post di blog sull'utilizzo del AWS Database Migration Service (DMS) per popolare il grafico di Neptune	Vedi Compilazione del grafico in Amazon Neptune da un database relazionale utilizzando Database AWS Migration Service (DMS) — Parte 2: Progettazione del modello del grafico delle proprietà, di Chris Smith.	22 ottobre 2020
Post di blog sull'utilizzo del AWS Database Migration Service (DMS) per popolare il grafico di Neptune	Vedi Compilazione del grafico in Amazon Neptune da un database relazionale utilizzando Database AWS Migration Service (DMS) — Parte 1: Setting the stage, di Chris Smith.	22 ottobre 2020
Versione del motore 1.0.4.0	A partire dal 12 ottobre 2020, viene implementata a livello generale la versione del motore 1.0.4.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.4.0.	12 ottobre 2020

Versione del motore 1.0.3.0.R2	A partire dal 12 ottobre 2020, viene implementata a livello generale la versione del motore 1.0.3.0.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.3.0.R2.	12 ottobre 2020
Versione del motore 1.0.2.2.R5	A partire dal 12 ottobre 2020, viene implementata a livello generale la versione del motore 1.0.2.2.R5. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.2.2.R5.	12 ottobre 2020
Post del blog sulla configurazione del VPC per le query federate SPARQL	Consulta Configure Amazon VPC for SPARQL 1.1 Federated Query with Amazon Neptune di Charles Ivie.	12 ottobre 2020
Post del blog sulla scrittura di un'eliminazione a cascata SPARQL	Consulta Write a cascading delete in SPARQL di Ora Lassila.	5 ottobre 2020

Post del blog sulla rappresentazione grafica AWS delle risorse con Neptune	Vedi Rappresenta graficamente AWS le tue risorse con Amazon Neptune, di Dave Bechberger.	28 settembre 2020
Post del blog sulla creazione di una mappatura terminologica MedDRA per la farmacovigilanza e la segnalazione degli eventi avversi utilizzando Neptune	Consulta Building Amazon Neptune based MedDRA terminology mapping for pharmacovigilance and adverse event reporting di Vaijayanti Joshi, Deven Atnoor, Ph.D, e Sudhanshu Malhotra.	24 settembre 2020
Post del blog sulla creazione di un grafo della conoscenza da un data warehouse utilizzando Neptune, per integrare l'intelligence commerciale	Consulta Complement Commercial Intelligence by Building a Knowledge Graph out of a Data Warehouse with Amazon Neptune di Shahria Hossain e Mikael Graindorge.	23 settembre 2020
Post del blog sul bilanciamento del carico con il client Gremlin per Neptune	Consulta Load balance graph queries using the Amazon Neptune Gremlin Client di Ian Robinson.	16 settembre 2020
Post del blog sulla personalizzazione digitale con un grafo di identità presso Cox Automotive	Consulta Cox Automotive scales digital personalization using an identity graph powered by Amazon Neptune di Carlos Rendon e Niraj Jetly.	16 settembre 2020
Post del blog sul filtro collaborativo sui dati di Yelp	Consulta Using collaborative filtering on Yelp data to build a recommendation system in Amazon Neptune di Chad Tindel.	8 settembre 2023

Post del blog sulla visualizzazione dei risultati delle query in Amazon Neptune	Consulta Visualize query results using the Amazon Neptune workbench di Kelvin Lawrence.	2 settembre 2020
Neptune ha rilasciato la visualizzazione grafo	Amazon Neptune ora offre numerose funzionalità di visualizzazione dei grafi nei notebook Jupyter in Neptune Workbench, oltre a varie nuove funzionalità che semplificano l'utilizzo dei notebook. Consulta Visualizzazione grafo.	12 agosto 2020
Neptune è stato lanciato in Sud America (San Paolo)	Amazon Neptune è ora disponibile in Sud America (San Paolo) (`sa-east-1`).	6 agosto 2020
Neptune è stato lanciato in Asia Pacifico (Hong Kong)	Amazon Neptune è ora disponibile in Asia Pacifico (Hong Kong) (`ap-east-1`).	6 agosto 2020
Versione del motore 1.0.3.0	A partire dal 03 agosto 2020, viene implementata a livello generale la versione del motore 1.0.3.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.3.0.	3 agosto 2020

Versione del motore 1.0.2.2.R4	A partire dal 23 luglio 2020, viene implementata a livello generale la versione del motore 1.0.2.2.R4. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.2.2.R4.	23 luglio 2020
Post del blog sul tracciamento automatico dei contatti di Zerobase con Amazon Neptune	Consulta Zerobase creates private, secure, and automated contact tracing using Amazon Neptune di David Harris e Aron Szanto.	13 luglio 2020
Neptune è stato lanciato negli Stati Uniti occidentali (California settentrionale)	Amazon Neptune è ora disponibile negli Stati Uniti occidentali (California settentrionale) (`us-west-1`).	9 luglio 2020
Amazon Neptune supporta il controllo degli accessi basato su tag	Ora puoi utilizzare i AWS tag nelle policy IAM per controllare l'accesso al tuo database Neptune. Consulta Controllo degli accessi basato su tag in Amazon Neptune.	7 luglio 2020

È ora disponibile uno strumento per il polling dei flussi Java	Amazon Neptune ora supporta una versione Java dello strumento per il polling dei flussi Lambda per i flussi Neptune oltre alla versione Python. Consulta Aggiunta di dettagli relativi allo stack consumer dei flussi Neptune che verranno creati.	6 luglio 2020
Post di blog sul Knowledge Graph sul AWS COVID-19	Vedi Creazione e interrogazione del grafico della conoscenza sul AWS COVID-19, di Ninad Kulkarni, Colby Wise, George Price e Miguel Romero.	1 luglio 2020
Versione del motore 1.0.1.1	A partire dal 26 giugno 2020, viene implementata a livello generale la versione del motore 1.0.1.1. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.1.1.	26 giugno 2020
Post del blog sulla migrazione da Blazegraph ad Amazon Neptune	Consulta Moving to the cloud: Migrazione di Blazegraph ad Amazon Neptune di Dave Bechberger.	25 giugno 2020

Post del blog sulla modifica dell'acquisizione dei dati da Neo4j ad Amazon Neptune	Consulta Change data capture from Neo4j to Amazon Neptune using Amazon Managed Streaming for Apache Kafka di Sanjeet Sahay.	22 giugno 2020
Post del blog su come Waves utilizza Amazon Neptune	Consulta How Waves runs user queries and recommend ations at scale with Amazon Neptune di Pavel Vasilyev.	16 giugno 2020
Versione del motore 1.0.1.2	A partire dal 10 giugno 2020, viene implementata a livello generale la versione del motore 1.0.1.2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazi oni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.1.2.	10 giugno 2020
Post del blog sulla creazione di un repository di conoscenze per i clienti	Consulta Building a customer 360 knowledge repository with Amazon Neptune and Amazon Redshift di Ram Bhandarkar.	9 giugno 2020
Post del blog su come Gunosy utilizza Amazon Neptune	Consulta How Gunosy built a comment feature in News Pass using Amazon Neptune di Yosuke Uchiyama.	8 giugno 2020

Post di AWS blog sul grafico della conoscenza del COVID-19	Vedi Creazione e interrogazione del grafico conoscitivo sul AWS COVID-19 di Ninad Kulkarni, Colby Wise, George Price e Miguel Romero.	2 giugno 2020
Post del blog sull'esplorazione della ricerca sul COVID-19 con Amazon Neptune	Consulta Exploring scientific research on COVID-19 with Amazon Neptune, Amazon Comprehend Medical, and the Tom Sawyer Graph Database Browser di George Price, Colby Wise, Miguel Romero e Ninad Kulkarni.	2 giugno 2020
Ora puoi caricare i dati in Neptune usando AWS DMS	Vedi Utilizzo di AWS Database Migration Service per caricare dati in Amazon Neptune da un altro data store.	1 giugno 2020
La versione del motore 1.0.2.0 è considerata obsoleta	La versione 1.0.2.0 del motore Amazon Neptune è ora obsoleta. I cluster in esecuzione su questa versione del motore verranno aggiornati automaticamente alla versione 1.0.2.1 durante la prima finestra di manutenzione successiva al 1° giugno 2020.	19 maggio 2020
Post del blog sulla creazione di un grafo di identità del cliente utilizzando Neptune	Vedere Creazione di un grafico di identità dei clienti con Amazon Neptune di Rajesh Wunnava e Taylor Riggan.	12 maggio 2020

Versione del motore 1.0.2.0.R3	A partire dal 05 maggio 2020, viene implementata a livello generale la versione del motore 1.0.2.0.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.2.0.R3.	5 maggio 2020
Versione del motore 1.0.2.1.R6	A partire dal 22 aprile 2020, viene implementata a livello generale la versione del motore 1.0.2.1.R6. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.2.1.R6.	22 aprile 2020
Post del blog sulla migrazione dei dati da Neo4j a Neptune	Vedere il post sulla migrazione di un database grafico Neo4j ad Amazon Neptune con un'utilità completamente automatizzata di Sanjeet Sahay.	13 aprile 2020

Post del blog sulla riduzione del costo dello sviluppo di applicazioni a grafo con Neptune	Vedere il post sulla riduzione del costo della creazione di app grafiche fino al 76% con le istanze T3 di Amazon Neptune di Karthik Bharathy e Brad Bebee.	9 aprile 2020
Neptune offre una classe di istanza espandibile T3	È ora possibile creare un'istanza espandibile T3 Amazon Neptune per scopi di sviluppo e test a costi convenienti. Vedere Classe di istanza espandibile T3 di Neptune.	8 aprile 2020
Versione del motore 1.0.2.2.R2	A partire dal 02 aprile 2020, viene implementata a livello generale la versione del motore 1.0.2.2.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.2.2.R2.	2 aprile 2020
Post del blog sulla rappresentazione grafica delle dipendenze degli investimenti in EDGAR	Vedere il post sulla dipendenza degli investimenti sulla grafica con Amazon Neptune di Lawrence Verdi.	17 marzo 2020
Neptune è stato lanciato in Europa (Parigi)	Amazon Neptune è ora disponibile in Europa (Parigi) (eu-west-3).	11 marzo 2020

Versione del motore 1.0.2.2	A partire dal 9 marzo 2020, viene implementata a livello generale la versione del motore 1.0.2.2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione. Per ulteriori informazioni su questa versione del motore, consulta Rilascio del motore Neptune 1.0.2.2.	9 marzo 2020
Arresto e riavvio di un cluster database	È ora possibile arrestare un cluster DB per 7 giorni utilizzando la console Neptune e successivamente riavviarlo quando è necessario. Per tutta la durata della sospensione di un cluster di database, vengono addebitati solo i costi per lo storage del cluster, gli snapshot manuali e lo storage di backup automatico, ma non per le ore delle istanze di database. Vedere Arresto e avvio di un cluster Amazon Neptune DB.	19 febbraio 2020

Video su un grafo social di Nike	Ascolta Todd Escalona mentre AWS parla con Marc Wangenheim, Senior Engineering Manager di Nike, su come l'azienda gestisce una serie di applicazioni tramite un social graph basato su Amazon Neptune. Vedi Nike: un grafico social su larga scala con Amazon Neptune.	11 febbraio 2020
I cluster Neptune possono ora essere configurati per richiedere connessioni SSL	Nelle regioni che supportano ancora le connessioni HTTP, SSL è ora attivato per impostazione predefinita in tutti i nuovi gruppi di parametri. Non ci sono modifiche ai gruppi di parametri esistenti, ma è possibile forzare i client a utilizzare SSL modificando il parametro `neptune_enforce_ssl` in 1. Vedere Crittografia in transito: Connessione a Nettuno tramite SSL/HTTPS per informazioni su come abilitare le connessioni HTTP per un cluster in un'area che le supporta ancora. Consulta Parametri che puoi usare per configurare Amazon Neptune per una descrizione dei parametri del cluster e dell'istanza.	10 febbraio 2020

Ora puoi specificare la versione del motore e la protezione dall'eliminazione nel modello di Neptune CloudFormation	Amazon Neptune ha aggiornato CloudFormation il modello per includere `AWS::Neptune::DBCluster.EngineVersion` un parametro che consente di specificare una particolare versione del motore per il nuovo cluster DB e `AWS::Neptune::DBCluster.DeletionProtection` un parametro che consente di attivare la protezione da eliminazione per tale cluster.	9 febbraio 2020
Deletion protection (Protezione da eliminazione)	Amazon Neptune offre la protezione dall'eliminazione per cluster e istanze di database. Finché la protezione da eliminazione è abilitata in un cluster o un'istanza database, non è possibile eliminarli. Consulta Impossibile eliminare un'istanza database se la protezione da eliminazione è abilitata.	20 gennaio 2020
Neptune è stato lanciato in Cina (Ningxia)	Amazon Neptune è ora disponibile in Cina (Ningxia) (cn-northwest-1).	15 gennaio 2020
Versione del motore 1.0.2.1.R4	La patch R4 per la versione del motore 1.0.2.1 è disponibile a livello generale. Per ulteriori informazioni, consulta Rilascio del motore Neptune 1.0.2.1.R4.	20 dicembre 2019

Versione del motore 1.0.2.1.R3	La patch R3 per la versione del motore 1.0.2.1 è disponibile a livello generale. Per ulteriori informazioni, consulta Rilascio del motore Neptune 1.0.2.1.R3.	12 dicembre 2019
Post del blog sull'utilizzo di Neptune per analizzare i feed dei social media	Vedi Analizzare i feed dei social media utilizzando Amazon Neptune.	27 novembre 2019
Versione del motore 1.0.2.1.R2	La patch R2 per la versione del motore 1.0.2.1 è disponibile a livello generale. Per ulteriori informazioni, consulta Rilascio del motore Neptune 1.0.2.1.R2.	25 novembre 2019
Versione del motore 1.0.2.1.R1	La versione 1.0.2.1.R1 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Rilascio del motore Neptune 1.0.2.1.	22 novembre 2019
Versione del motore 1.0.2.0.R2	La patch R2 per la versione del motore 1.0.2.0 è disponibile a livello generale. Per ulteriori informazioni, consulta Rilascio del motore Neptune 1.0.2.0.R2.	21 novembre 2019
Post del blog su sessioni e workshop su Neptune all'evento re:Invent 2019	Consulta la tua guida alle sessioni, ai workshop e alle conferenze su Amazon Neptune a re:Invent 2019. AWS	20 novembre 2019

Versione del motore 1.0.2.0.R1	La versione 1.0.2.0.R1 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Rilascio del motore Neptune 1.0.2.0.	8 Novembre 2019
Post del blog sull'acquisizione di modifiche ai grafi mediante i flussi Neptune	Consultare Capture Graph Changes using Neptune Streams (Acquisizione delle modifiche ai grafici mediante i flussi di Neptune).	6 novembre 2019
Versione del motore 1.0.1.0.200502.0	La versione 1.0.1.0.200502.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200502.0.	31 ottobre 2019
Neptune è stato lanciato in Medio Oriente (Bahrein)	Amazon Neptune è ora disponibile in Medio Oriente (Bahrein) (me-south-1).	30 ottobre 2019
Neptune è stato lanciato in Canada (Centrale)	Amazon Neptune è ora disponibile in Canada (Centrale) (ca-central-1).	30 ottobre 2019
Post del blog sulla nuova funzionalità Flussi SPARQL di Neptune e sul supporto per le query federate SPARQL	Consulta Amazon Neptune releases Streams, SPARQL federated query for graphs and more.	17 ottobre 2019

Versione del motore 1.0.1.0.200463.0	La versione 1.0.1.0.200463.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200463.0.	15 ottobre 2019
Versione del motore 1.0.1.0.200457.0	La versione 1.0.1.0.200457.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200457.0.	19 settembre 2019
Post del blog sulla nuova funzionalità SPARQL Explain di Neptune	Consulta Using SPARQL explain to understand query execution in Amazon Neptune.	17 settembre 2019
Post di blog sul supporto di Neptune per la versione 3.4 TinkerPop	Vedi Amazon Neptune ora TinkerPop supporta le funzionalità 3.4.	6 settembre 2019
Post di blog sull'utilizzo di PyTorch Neptune con Amazon SageMaker	Vedi Un'esperienza personalizzata con shop-by-style " PyTorch su Amazon SageMaker e Amazon Neptune.	22 agosto 2019
Post di blog sull'uso di Neptune con AWS AppSync Amazon Elasticache	Vedi Integrazione di fonti di dati alternative con AWS AppSync: Amazon ElastiCache Neptune e Amazon.	22 agosto 2019
Neptune è stato lanciato AWS GovCloud nel (Stati Uniti orientali)	Amazon Neptune è ora disponibile AWS GovCloud in (Stati Uniti orientali) (-1). us-gov-east	21 agosto 2019

Neptune è stato lanciato AWS GovCloud nel (Stati Uniti occidentali)	Amazon Neptune è ora disponibile AWS GovCloud in (Stati Uniti occidentali) (-1). us-gov-west	14 agosto 2019
Versione del motore 1.0.1.0.200369.0	La versione 1.0.1.0.200369.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200369.0.	13 agosto 2019
Versione del motore 1.0.1.0.200366.0	La versione 1.0.1.0.200366.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200366.0.	26 luglio 2019
Post di blog sull'utilizzo di PyTorch Neptune con Amazon SageMaker	Vedi Un'esperienza personalizzata con shop-by-style " PyTorch su Amazon SageMaker e Amazon Neptune.	3 luglio 2019
Versione del motore 1.0.1.0.200348.0	La versione 1.0.1.0.200348.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200348.0.	2 luglio 2019
Neptune è stato lanciato in Europa (Stoccolma)	Amazon Neptune è ora disponibile in Europa (Stoccolma) (eu-north-1).	27 giugno 2019

Neptune può ora pubblicare i log di controllo su Logs CloudWatch	Per ulteriori informazioni, consulta Pubblicazione dei log di Neptune su Amazon Logs. CloudWatch	18 giugno 2019
Versione del motore 1.0.1.0.200310.0	La versione 1.0.1.0.200310.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200310.0.	12 giugno 2019
Post sul blog che parla di LifeOmic JupiterOne	Scopri LifeOmicHow JupiterOne semplifica le operazioni di sicurezza e conformità con Amazon Neptune.	2 maggio 2019
Neptune è stato lanciato in Asia Pacifico (Seoul)	Amazon Neptune è ora disponibile in Asia Pacifico (Seoul) (ap-northeast-2).	1 maggio 2019
Versione del motore 1.0.1.0.200296.0	La versione 1.0.1.0.200296.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200296.0.	1 maggio 2019
Neptune è stato lanciato in Asia Pacifico (Mumbai)	Amazon Neptune è ora disponibile in Asia Pacifico (Mumbai) (ap-south-1).	6 marzo 2019
Post del blog sugli hint di query Gremlin	Consulta Introducing Gremlin query hints for Amazon Neptune.	26 febbraio 2019

Neptune è stato lanciato in Asia Pacifico (Tokyo)	Amazon Neptune è ora disponibile in Asia Pacifico (Tokyo) (ap-northeast-1).	23 gennaio 2019
AWS CloudFormation modello per creare una AWS Lambda funzione per accedere a Neptune	È stata aggiornata la sezione introduttiva e aggiunto un AWS CloudFormation modello per creare una funzione Lambda da utilizzare con Neptune. Per ulteriori informazioni, consulta Nozioni di base su Neptune.	23 gennaio 2019
Versione del motore 1.0.1.0.200267.0	La versione 1.0.1.0.200267.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200267.0.	21 gennaio 2019
Neptune è stato lanciato in Asia Pacifico (Sydney)	Amazon Neptune è ora disponibile in Asia Pacifico (Sydney) (ap-southeast-2).	9 gennaio 2019
Post del blog sull'utilizzo di Metaphactory	Consulta Exploring Knowledge Graphs on Amazon Neptune Using Metaphactory.	9 gennaio 2019
Neptune è stato lanciato in Asia Pacifico (Singapore)	Amazon Neptune è ora disponibile in Asia Pacifico (Singapore) (ap-southeast-1).	13 dicembre 2018
Versione del motore 1.0.1.0.200264.0	La versione 1.0.1.0.200264.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200264.0.	19 novembre 2018

Supporto di SSL per Amazon Neptune	Neptune ora supporta le connessioni SSL.	19 novembre 2018
Argomenti degli errori consolidati	Tutte le informazioni sul codice e sul messaggio di errore sono ora disponibili in un singolo argomento.	15 novembre 2018
Aggiornamento dell'argomento sulle nozioni di base	Argomento sulle nozioni di base aggiornato con collegamenti aggiuntivi e documentazione riorganizzata.	14 novembre 2018
Versione del motore 1.0.1.0.200258.0	La versione 1.0.1.0.200258.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200258.0.	8 novembre 2018
Neptune è stato lanciato in Europa (Francoforte)	Amazon Neptune è ora disponibile in Europa (Francoforte) (eu-central-1).	7 novembre 2018
Post del blog n. 1 in una serie	Consulta Let Me Graph That For You – Part 1 – Air Routes.	7 novembre 2018
Post di blog sull'uso di Amazon SageMaker Jupyter Notebooks	Consulta Analizza i grafici di Amazon Neptune con i notebook Amazon Jupyter. SageMaker	1 novembre 2018
Versione del motore 1.0.1.0.200255.0	La versione 1.0.1.0.200255.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200255.0.	29 ottobre 2018

Neptune è stato lanciato in Europa (Londra)	Amazon Neptune è ora disponibile in Europa (Londra) (eu-west-2).	3 ottobre 2018
Versione del motore 1.0.1.0.200237.0	La versione 1.0.1.0.200237.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200237.0.	6 settembre 2018
Versione del motore 1.0.1.0.200236.0	La versione 1.0.1.0.200236.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200236.0.	24 luglio 2018
Versione del motore 1.0.1.0.200233.0	La versione 1.0.1.0.200233.0 del motore Amazon Neptune è disponibile a livello generale. Per ulteriori informazioni, consulta Aggiornamento 1.0.1.0.200233.0.	22 giugno 2018
Nuova guida Quick Start di Neptune	Guida rapida aggiornata con Gremlin Console e tutorial. AWS CloudFormation Per ulteriori informazioni, consulta Amazon Neptune Quick Start Using. AWS CloudFormation	19 giugno 2018

Rilascio iniziale di Amazon Neptune	Si tratta della versione iniziale della Guida per l'utente di Neptune. Consulta anche il post del blog di rilascio, Amazon Neptune Generally Available.	30 maggio 2018
Post del blog introduttivo su Neptune	Consulta Amazon Neptune – A Fully Managed Graph Database Service.	29 novembre 2017

Nozioni di base su Amazon Neptune

Amazon Neptune è un servizio di database a grafo completamente gestito e scalabile per gestire miliardi di relazioni e consente di eseguire query su di esse con una latenza di millisecondi, a un costo contenuto per tale tipo di capacità.

Per informazioni più dettagliate su Neptune, consulta Panoramica delle funzionalità di Amazon Neptune.

Se hai già familiarità con i grafi, passa a Utilizzo di notebook per grafi. Oppure, se desideri creare subito un database Neptune, consulta Utilizzo di uno AWS CloudFormation stack per creare un cluster Neptune DB.

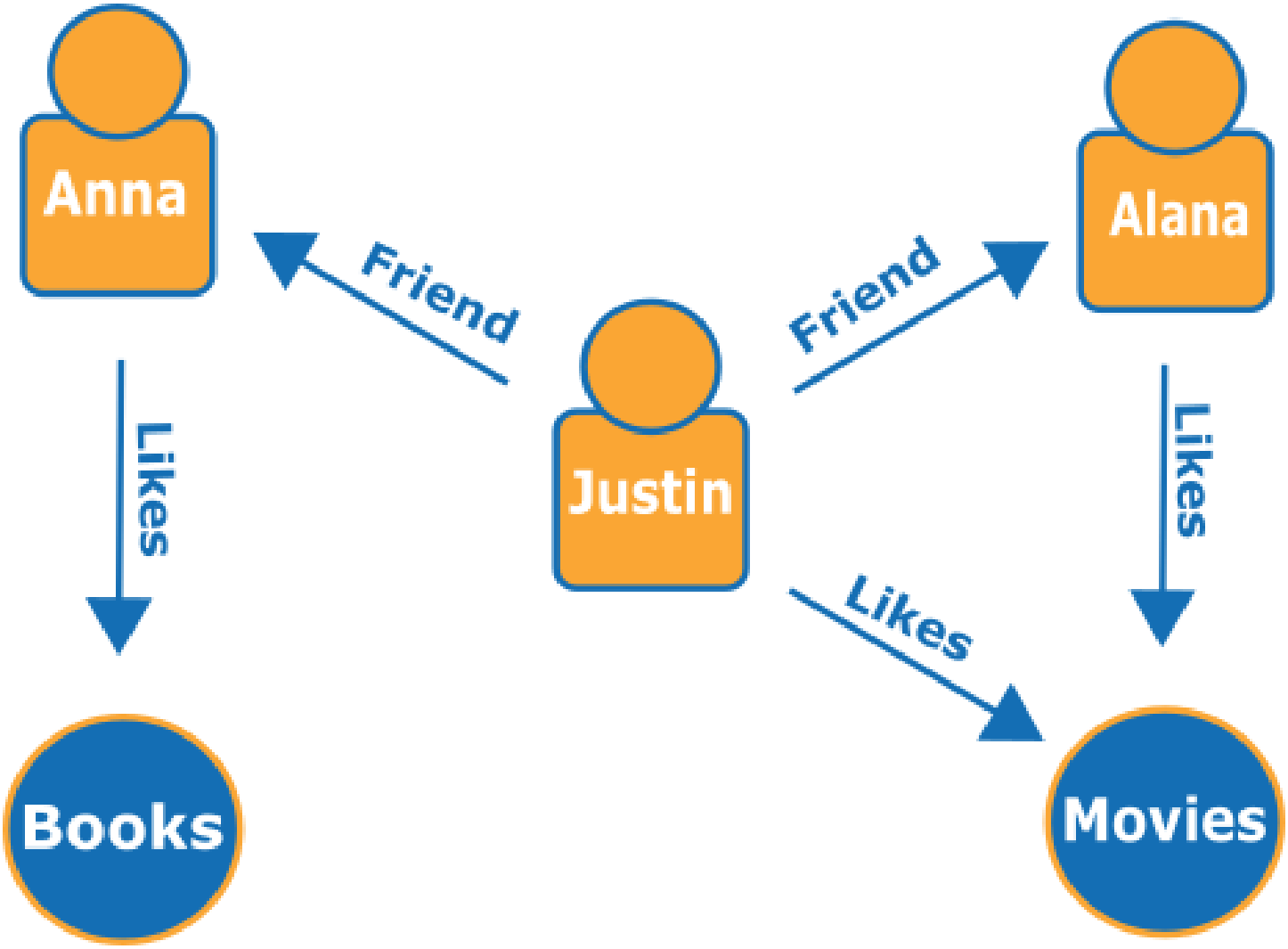
In caso contrario, puoi approfondire i concetti relativi ai database a grafo prima di iniziare.

Cos'è esattamente un database a grafo?

I database a grafo sono ottimizzati per archiviare le relazioni tra gli elementi di dati ed eseguire query su di esse.

Archiviano gli elementi di dati stessi come vertici del grafo e le relazioni tra di essi come archi. Ogni arco è di un tipo specifico ed è diretto da un vertice (l'inizio) all'altro (la fine). Le relazioni possono essere denominate predicati oltre che archi e i vertici vengono talvolta chiamati anche nodi. Nei cosiddetti grafi di proprietà, sia ai vertici che agli archi possono essere associate anche proprietà aggiuntive.

Ecco un piccolo grafo che rappresenta amici e hobby in un social network:



Gli archi sono rappresentati da frecce denominate e i vertici rappresentano persone e hobby specifici a cui si connettono.

Un semplice attraversamento di questo grafo ti consente di conoscere quali sono le cose che gli amici di Justin amano.

Perché usare un database a grafo?

Un database a grafo è la scelta migliore nei casi in cui le connessioni o relazioni tra entità sono al centro dei dati che si vogliono modellare.

Innanzitutto, modellare le interconnessioni di dati sotto forma di grafo e quindi scrivere query complesse che estraggono informazioni reali dal grafo è facile.

Lo sviluppo di un'applicazione equivalente utilizzando un database relazionale richiede la creazione di molte tabelle con più chiavi esterne e quindi la scrittura di query SQL annidate e join complessi. Questo approccio non solo diventa rapidamente macchinoso dal punto di vista della scrittura del codice, ma le sue prestazioni peggiorano velocemente con l'aumentare della quantità di dati.

Al contrario, un database a grafo come Neptune può eseguire query sulle relazioni tra miliardi di vertici senza rimanere bloccato.

Cosa è possibile fare con un database a grafo?

I grafi possono rappresentare le interrelazioni tra le entità del mondo reale in molti modi, in termini di azioni, proprietà, parentela, scelte di acquisto, legami personali, legami familiari e così via.

Ecco alcune delle aree più comuni in cui vengono utilizzati i database a grafo:

- Grafi della conoscenza: i grafi della conoscenza consentono di organizzare tutti i tipi di informazioni connesse ed eseguire query su di esse per rispondere a domande generali. Utilizzando un grafo della conoscenza, puoi aggiungere informazioni tematiche a cataloghi di prodotti e modellare informazioni eterogenee come quelle contenute in Wikidata.

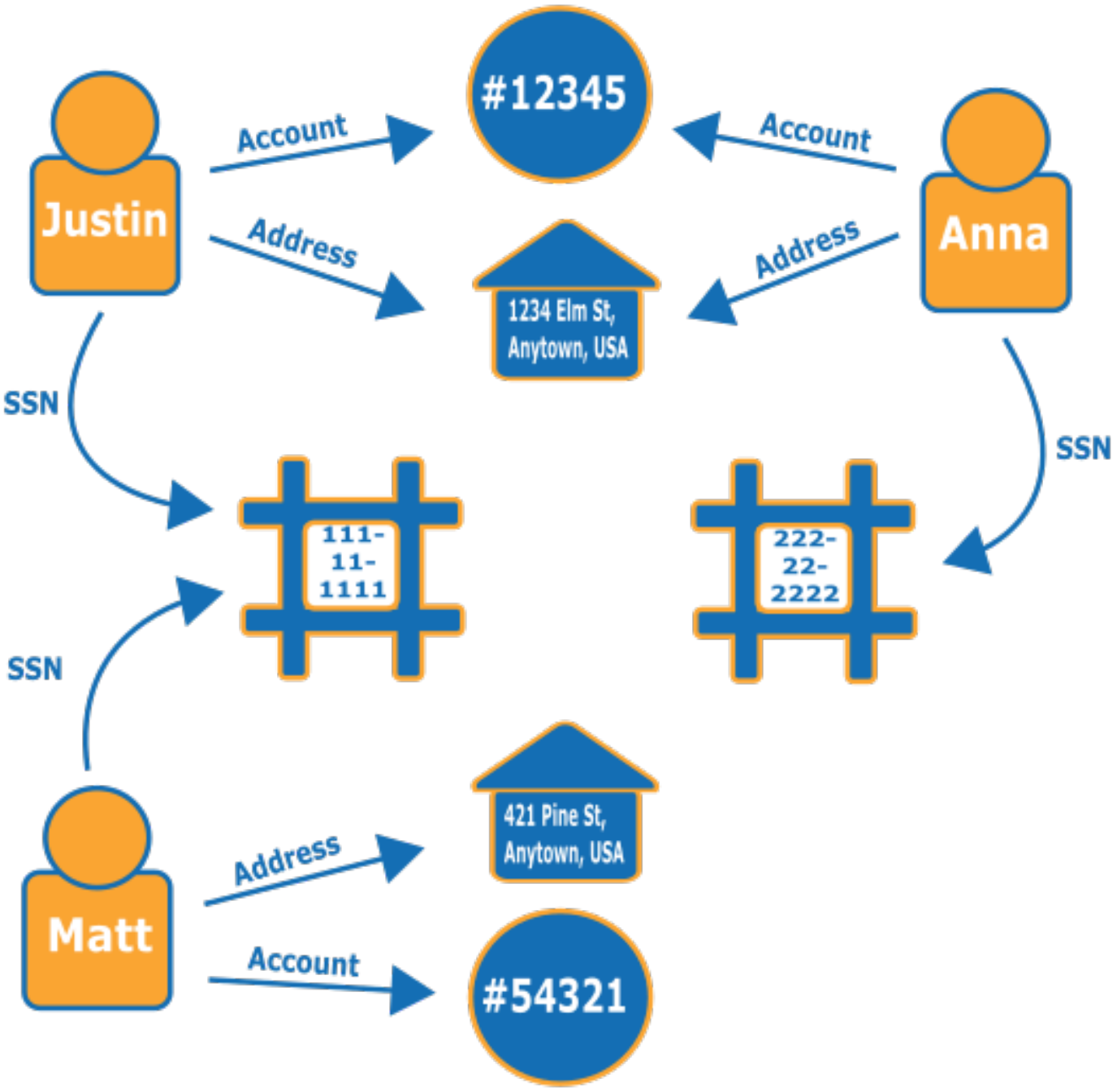
  Per saperne di più su come funzionano i grafi della conoscenza e su dove vengono utilizzati, consulta Grafi della conoscenza in AWS.
- Grafi di identità: in un database a grafo, puoi archiviare le relazioni tra categorie di informazioni, come interessi dei clienti, amici e cronologia degli acquisti, e quindi eseguire query su tali dati per formulare raccomandazioni personalizzate e pertinenti.

  Ad esempio, puoi utilizzare un database a grafo per raccomandare a un utente prodotti da acquistare in base ai prodotti acquistati da altri che seguono lo stesso sport e hanno una cronologia di acquisto simile. Oppure puoi identificare le persone che hanno un amico in comune ma non si conoscono ancora e suggerire amicizie.

  I grafi di questo tipo sono noti come grafi di identità e sono ampiamente utilizzati per personalizzare le interazioni con gli utenti. Per saperne di più, consulta Grafi di identità in AWS. Per creare il tuo grafo di identità, puoi iniziare con l'esempio Identity Graph Using Amazon Neptune.
- Grafi di rilevamento delle frodi: si tratta di un utilizzo comune per i database a grafo. Possono aiutarti a monitorare gli acquisti con carta di credito e le località di acquisto per individuare utilizzi insoliti o per rilevare che un acquirente sta tentando di utilizzare lo stesso indirizzo e-mail e la stessa carta di credito usati in un caso di frode noto. Ti permettono inoltre di verificare se esistono

sono più persone associate a un indirizzo e-mail personale o più persone in luoghi fisici diversi che condividono lo stesso indirizzo IP.

Considera il grafo seguente. Il grafo seguente mostra le relazioni tra tre persone e le informazioni relative alla loro identità. Ogni persona dispone di un indirizzo, un conto corrente e un social security number. Tuttavia, si può notare che Matt e Justin condividono lo stesso codice SSN, fatto irregolare che indica una possibile frode da parte di uno di essi. Una query su un grafo di rilevamento delle frodi può rivelare connessioni di questo tipo affinché possano essere esaminate.



Per saperne di più su come funzionano i grafi di rilevamento di attività fraudolente e su dove vengono utilizzati, consulta Fraud Graphs on AWS.

- Social network: le applicazioni di social network costituiscono una delle aree principali e più comuni in cui vengono utilizzati i database a grafo.

Supponiamo, ad esempio, che tu voglia creare un social feed in un sito Web. Puoi utilizzare facilmente un database a grafo sul back-end per fornire agli utenti risultati che riflettano gli ultimi aggiornamenti relativi a familiari, amici, persone con aggiornamenti a cui si è risposto con "Mi piace" e persone che vivono nelle vicinanze.

- Indicazioni stradali: un grafo può aiutare a individuare il percorso migliore da un punto di partenza a una destinazione, in base al traffico attuale e ai modelli di traffico tipici.
- Logistica: i grafi consentono di identificare il modo più efficiente di utilizzare le risorse di spedizione e distribuzione disponibili per soddisfare le esigenze dei clienti.
- Diagnostica: i grafi possono rappresentare alberi diagnostici complessi su cui si possono eseguire query per identificare l'origine dei problemi e degli errori osservati.
- Ricerca scientifica: con un database a grafo, è possibile creare applicazioni per archiviare ed esplorare i dati scientifici e persino le informazioni mediche sensibili utilizzando la crittografia dei dati inattivi. È ad esempio possibile archiviare i modelli di interazioni tra geni e patologie. È possibile cercare modelli di grafo all'interno di percorsi di proteine per trovare altri geni che potrebbero essere associati a una patologia. Puoi modellare composti chimici come grafi ed eseguire query per individuare modelli nelle strutture molecolari. È possibile correlare i dati dei pazienti contenuti nelle cartelle cliniche in sistemi diversi. Puoi organizzare in modo topico le pubblicazioni delle ricerche per trovare informazioni rilevanti in modo rapido.
- Regole normative: è possibile archiviare requisiti normativi complessi sotto forma di grafi ed eseguire query su di essi per rilevare situazioni in cui potrebbero risultare validi per le operazioni aziendali quotidiane.
- Topologia ed eventi di rete: un database a grafo può aiutarti a gestire e proteggere una rete IT. Quando si archivia la topologia di rete come grafo, è anche possibile archiviare ed elaborare molti tipi diversi di eventi sulla rete. È possibile rispondere a varie domande, ad esempio quanti host eseguono un'applicazione specifica. È possibile eseguire query sui modelli che potrebbero dimostrare che un determinato host è stato compromesso da un programma dannoso e cercare i dati delle connessioni per risalire all'host originale che ha scaricato il programma.

Come si eseguono le query su un grafo?

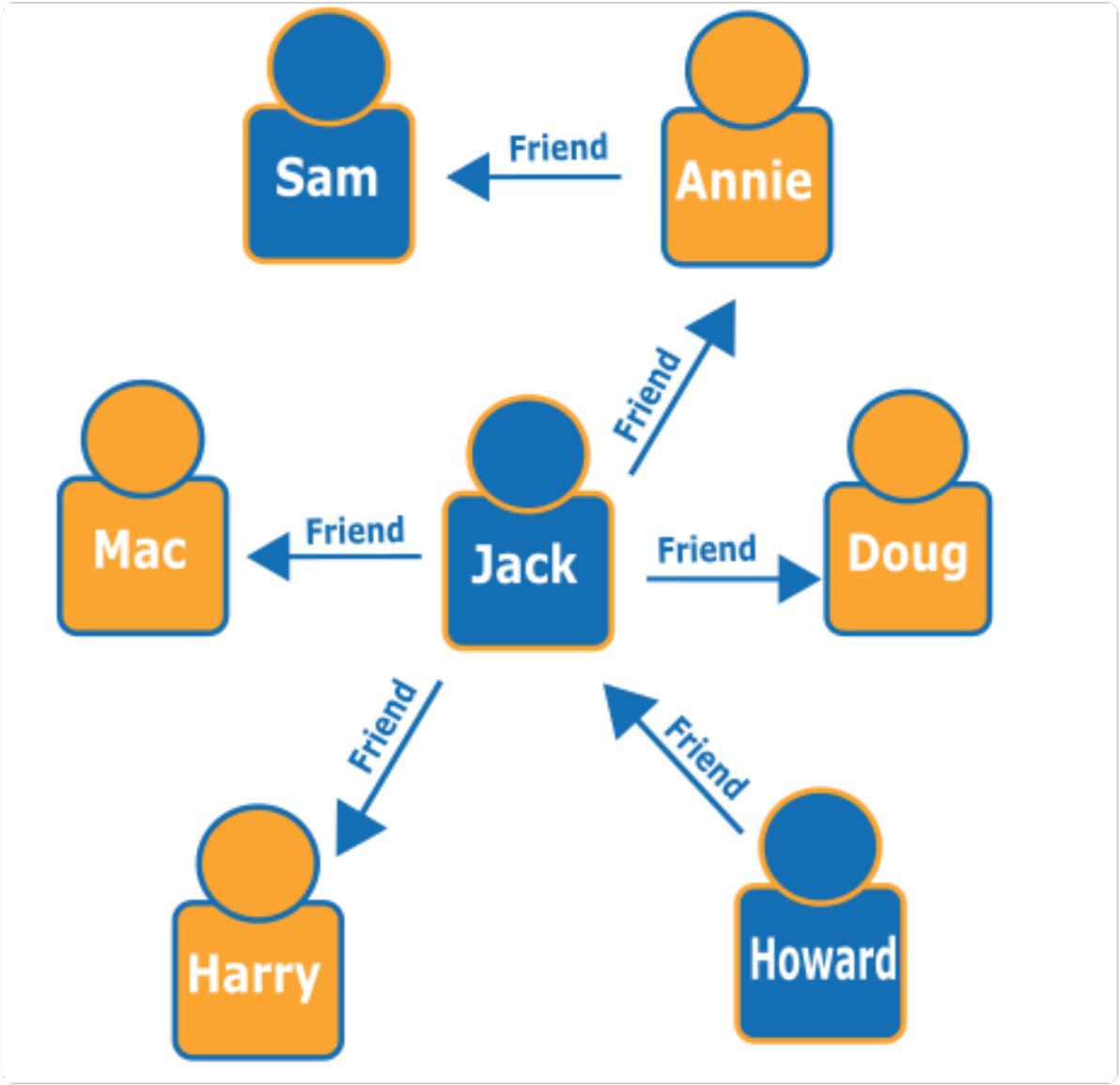
Neptune supporta tre linguaggi di query speciali progettati per l'esecuzione di query sui dati dei grafi di tipi diversi. È possibile utilizzare questi linguaggi per aggiungere, modificare, eliminare ed eseguire query sui dati in un database a grafo Neptune:

- Gremlin è un linguaggio di attraversamento del grafo per grafi di proprietà. Una query in Gremlin è un attraversamento composto da passaggi discreti, ognuno dei quali segue un arco fino a un nodo. Per ulteriori informazioni, consulta la documentazione di Gremlin su Apache TinkerPop3.

  L'implementazione di Gremlin in Neptune differisce da altre implementazioni, soprattutto se utilizzi Gremlin-Groovy (query Gremlin inviate come testo serializzato). Per ulteriori informazioni, consulta Conformità agli standard Gremlin in Amazon Neptune.
- openCypher: openCypher è un linguaggio di query dichiarativo per grafi di proprietà che è stato sviluppato originariamente da Neo4j, per poi diventare open source nel 2015, e che ha contribuito al progetto openCypher con una licenza open source Apache 2. Consulta i documenti Cypher Query Language Reference (versione 9) per le specifiche del linguaggio e Cypher Style Guide per ulteriori informazioni.
- SPARQL è un linguaggio di query dichiarativo per i dati RDF basato sul tipo di corrispondenza del modello di grafo standardizzato dal World Wide Web Consortium (W3C) e descritto in SPARQL 1.1 Overview e nella specifica SPARQL 1.1 Query Language. Consulta Conformità agli standard SPARQL in Amazon Neptune per dettagli specifici sull'implementazione Neptune di SPARQL.

Esempi di corrispondenza delle query Gremlin e SPARQL

Considerati i seguenti grafi di persone (nodi) e le relative relazioni (archi), puoi scoprire che "gli amici di amici" di una persona specifica sono, ad esempio, amici di amici di Howard.



Guardando il grafico, è possibile verificare che Howard ha un amico, Jack e Jack ha quattro amici: Annie, Harry, Doug e Mac. Questo è un esempio semplice con un grafo semplice, ma questi tipi di query possono scalare in termini di complessità, set di dati e dimensione dei risultati.

Di seguito è riportata una query di attraversamento Gremlin che restituisce i nomi degli amici degli amici di Howard.

```
g.V().has('name', 'Howard').out('friend').out('friend').values('name')
```

Di seguito è riportata una query SPARQL che restituisce i nomi degli amici degli amici di Howard.

```
prefix : <#>

  select ?names where {
    ?howard :name "Howard" .
    ?howard :friend/:friend/:name ?names .
  }
```

> **Note**
>
> Ogni parte di qualsiasi tripla in Resource Description Framework (RDF) dispone di un URI associato. In questo esempio, il prefisso dell'URI è intenzionalmente breve.

Segui un corso online sull'utilizzo di Amazon Neptune

Se ti piace imparare con i video, AWS offre corsi online in AWSOnline Tech Talks per migliorare le tue competenze al ritmo che preferisci. Un corso introduttivo sui database a grafo è::

Graph Database introduction, deep-dive and demo with Amazon Neptune.

Approfondimenti sull'architettura di riferimento dei grafi

Mentre rifletti sui problemi che un database a grafo più aiutarti a risolvere e su come affrontarli, uno dei migliori punti di partenza è il progetto GitHub per le architetture di riferimento dei grafi Neptune.

Qui puoi trovare descrizioni dettagliate dei tipi di carichi di lavoro dei grafi e tre sezioni per aiutarti a progettare un database a grafo efficace:

- Modelli di dati e linguaggi di query: questa sezione illustra le differenze tra Gremlin e SPARQL e come scegliere il linguaggio più adatto.
- Modellazione dei dati dei grafi: si tratta di una discussione approfondita su come prendere decisioni sulla modellazione dei dati dei grafi, incluse procedure dettagliate sulla modellazione dei grafi di proprietà con Gremlin e sulla modellazione RDF con SPARQL.
- Conversione di altri modelli di dati in un modello a grafo: fornisce informazioni su come convertire un modello di dati relazionale in un modello a grafo.

Sono inoltre disponibili tre sezioni che illustrano i passaggi specifici per l'utilizzo di Neptune:

- Connessione ad Amazon Neptune da client esterni al VPC di Neptune: questa sezione mostra numerose opzioni per la connessione a Neptune dall'esterno del VPC in cui si trova il cluster database.
- Accesso ad Amazon Neptune da funzioni AWS Lambda: illustra come connettersi in modo affidabile a Neptune dalle funzioni Lambda.
- Scrittura su Amazon Neptune da un flusso di dati Amazon Kinesis: questa sezione fornisce informazioni su come gestire scenari con velocità effettiva di scrittura elevata con Neptune.

Utilizzo di notebook per grafi Neptune per iniziare rapidamente

Non è necessario usare i notebook per grafi Neptune per utilizzare un grafo Neptune, quindi se lo desideri, puoi procedere e creare subito un nuovo database Neptune utilizzando un modello AWS CloudFormation.

Allo stesso tempo, Neptune Workbench offre un ambiente di sviluppo interattivo (IDE) che può aumentare la produttività durante lo sviluppo di applicazioni a grafo sia per gli utenti che non hanno familiarità con i grafi e che desiderano apprendere e sperimentare, sia per gli utenti esperti che vogliono perfezionare le proprie query.

Neptune fornisce notebook Jupyter e JupyterLab nel progetto open source notebook per grafi Neptune su GitHub e in Neptune Workbench. Questi notebook offrono tutorial per applicazioni di esempio e frammenti di codice in un ambiente di sviluppo interattivo in cui è possibile apprendere i concetti relativi alla tecnologia dei grafi e a Neptune. Puoi usarli per visualizzare le procedure di installazione, configurazione, popolamento ed esecuzione di query sui grafi con diversi linguaggi di query, diversi set di dati e persino diversi database sul back-end.

È possibile ospitare questi notebook in vari modi:

- Neptune Workbench ti consente di eseguire notebook Jupyter in un ambiente completamente gestito, ospitato in Amazon SageMaker, e carica automaticamente l'ultimo rilascio del progetto notebook per i grafi Neptune. Configurare Workbench nella console Neptune quando si crea un nuovo database Neptune è facile.
- Puoi anche installare Jupyter in locale. Potrai così eseguire i notebook dal laptop, collegato a Neptune o a un'istanza locale di uno dei database a grafo open source. In quest'ultimo caso, avrai la possibilità sperimentare con la tecnologia dei grafi quanto desideri prima di spendere un

centesimo. Quindi, quando sei pronto, puoi passare senza problemi all'ambiente di produzione gestito offerto da Neptune.

Utilizzo Neptune Workbench per ospitare i notebook Neptune

Per iniziare, Neptune offre tipi di istanze T3 e T4g a meno di 0,10 USD all'ora. Le risorse Workbench vengono fatturate attraverso Amazon SageMaker, separatamente dalla fatturazione di Neptune. Consulta la pagina dei prezzi di Neptune. I notebook Jupyter e JupyterLab creati in Neptune Workbench utilizzano tutti un ambiente Amazon Linux 2 e JupyterLab 3. Per ulteriori informazioni sul supporto per notebook JupyterLab, consulta la documentazione di Amazon SageMaker.

Esistono due modi per creare un notebook Jupyter o JupyterLab con Neptune Workbench nella AWS Management Console:

- Usa il menu Configurazione Notebook durante la creazione di un nuovo cluster database Neptune. A tale scopo, segui la procedura descritta in Avvio di un cluster database Neptune mediante la console AWS Management Console.
- Usa il menu Notebook nel riquadro di navigazione a sinistra dopo che il tuo cluster database è già stato creato. A tale scopo, attenersi alla procedura riportata di seguito.

Per creare un notebook Jupyter o JupyterLab utilizzando il menu Notebook

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione sinistro, scegliere Notebooks (Notebook).
3. Seleziona Create Notebook (Crea notebook).
4. Nell'elenco Cluster scegli il tuo cluster database Neptune. Se non si dispone ancora di un cluster DB, scegliere Create cluster (Crea cluster) per crearne uno.
5. Seleziona un tipo di istanza Notebook.
6. Fornire un nome al notebook e, facoltativamente, una descrizione.
7. A meno che non sia già stato creato un ruolo AWS Identity and Access Management (IAM) per i notebook, scegli Crea un ruolo IAM e immetti un nome di ruolo IAM.

> **Note**
>
> Se scegli di riutilizzare un ruolo IAM, creato per un notebook precedente, la policy del ruolo deve contenere le autorizzazioni corrette per accedere al cluster database Neptune in uso. Puoi verificare controllando che i componenti nel nome ARN della risorsa nell'azione `neptune-db:*` corrispondano a quel cluster. Le autorizzazioni configurate in modo errato provocano errori di connessione quando si tenta di eseguire i comandi magic dei notebook.

8. Seleziona Create Notebook (Crea notebook). Il processo di creazione può richiedere da 5 a 10 minuti prima che tutto sia pronto.
9. Dopo aver creato il notebook, selezionalo e scegli Apri Jupyter o Apri JupyterLab.

La console può creare un ruolo AWS Identity and Access Management (IAM) per i notebook o puoi crearne uno manualmente. Il criterio per questo ruolo dovrebbe includere quanto segue:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::aws-neptune-notebook-(AWS region)",
        "arn:aws:s3:::aws-neptune-notebook-(AWS region)/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": [
        "arn:aws:neptune-db:(AWS region):(AWS account ID):(Neptune resource ID)/*"
      ]
    }
  ]
}
```

Tieni presente che la seconda istruzione nella policy precedente elenca uno o più ID risorsa del cluster Neptune.

Inoltre, il ruolo dovrebbe stabilire la seguente relazione di trust:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Ancora una volta, la preparazione può richiedere da 5 a 10 minuti.

È possibile configurare il nuovo notebook per funzionare con Neptune ML, come spiegato in Configurazione manuale di un notebook Neptune per Neptune ML.

Utilizzo di Python per connettere un notebook SageMaker generico a Neptune

Connettere un notebook a Neptune è facile se hai installato i comandi magic di Neptune, ma è anche possibile connettere un notebook SageMaker a Neptune usando Python, anche se non usi un notebook Neptune.

Procedura da eseguire per connettersi a Neptune in una cella di notebook SageMaker

1. Installa il client Gremlin Python:

   ```
   !pip install gremlinpython
   ```

   I notebook Neptune installano automaticamente il client Gremlin Python, quindi questo passaggio è necessario solo se utilizzi un notebook SageMaker semplice.
2. Per connetterti ed eseguire una query Gremlin, scrivi il codice seguente:

   ```
   from gremlin_python import statics
   from gremlin_python.structure.graph import Graph
   ```

```
from gremlin_python.process.graph_traversal import __
from gremlin_python.process.strategies import *
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection
from gremlin_python.driver.aiohttp.transport import AiohttpTransport
from gremlin_python.process.traversal import *
import os

port = 8182
server = '(your server endpoint)'

endpoint = f'wss://{server}:{port}/gremlin'

graph=Graph()

connection = DriverRemoteConnection(endpoint,'g',

 transport_factory=lambda:AiohttpTransport(call_from_event_loop=True))

g = graph.traversal().withRemote(connection)

results = (g.V().hasLabel('airport')
                .sample(10)
                .order()
                .by('code')
                .local(__.values('code','city').fold())
                .toList())

# Print the results in a tabular form with a row index
for i,c in enumerate(results,1):
    print("%3d %4s %s" % (i,c[0],c[1]))

connection.close()
```

Note

Se stai usando una versione del client Gremlin Python precedente alla 3.5.0, questa riga:

```
connection = DriverRemoteConnection(endpoint,'g',

 transport_factory=lambda:AiohttpTransport(call_from_event_loop=True))
```

Sarà solo:

```
connection = DriverRemoteConnection(endpoint,'g')
```

Abilitazione dei log di CloudWatch nei notebook Neptune

I log di CloudWatch ora sono abilitati per impostazione predefinita per i notebook Neptune. Se usi notebook meno recente che non genera i log di CloudWatch, segui questa procedura per abilitarli manualmente:

1. Accedi alla AWS Management Console e apri la console SageMaker.
2. Nel riquadro di navigazione a sinistra scegli Notebook, quindi Istanze notebook. Cerca il nome del notebook Neptune per il quale desideri abilitare i log.
3. Vai alla pagina dei dettagli selezionando il nome dell'istanza notebook.
4. Se l'istanza notebook è in esecuzione, seleziona il pulsante Arresta in alto a destra nella pagina dei dettagli del notebook.
5. In Autorizzazioni e crittografia è disponibile un campo per ARN del ruolo IAM. Seleziona il collegamento in questo campo per passare al ruolo IAM con cui viene eseguita questa istanza notebook.
6. Crea la policy seguente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DeleteLogDelivery",
        "logs:Describe*",
        "logs:GetLogDelivery",
        "logs:GetLogEvents",
        "logs:ListLogDeliveries",
        "logs:PutLogEvents",
        "logs:PutResourcePolicy",
        "logs:UpdateLogDelivery"
      ],
```

```
      "Resource": "*"
    }
  ]
}
```

7. Salva questa nuova policy e collegala al ruolo IAM trovato nel passaggio 4.
8. Fai clic su Avvia in alto a destra nella pagina dei dettagli dell'istanza notebook di SageMaker.
9. Quando inizia il flusso dei log, visualizzerai un collegamento Visualizza log sotto il campo denominato Configurazione del ciclo di vita nella parte inferiore sinistra della sezione Impostazioni dell'istanza notebook della pagina dei dettagli.

Se un notebook non si avvia, nella pagina dei dettagli del notebook sulla console SageMaker verrà visualizzato un messaggio che indica che l'avvio dell'istanza notebook ha richiesto più di 5 minuti. I log di CloudWatch relativi a questo problema sono disponibili con questo nome:

```
(your-notebook-name)/LifecycleConfigOnStart
```

Configurazione dei notebook per grafi nel computer locale

Il progetto graph-notebook contiene le istruzioni per configurare i notebook Neptune nel computer locale:

- Prerequisiti
- Installazione di Jupyter e JupyterLab
- Connessione a un database a grafo

È possibile connettere i notebook locali a un cluster database Neptune o a un'istanza locale o remota di un database a grafo open source.

Utilizzo dei notebook Neptune con i cluster Neptune

Se ti connetti a un cluster Neptune sul back-end, puoi eseguire i notebook in Amazon SageMaker. La connessione a Neptune da SageMaker è più pratica rispetto a un'installazione locale dei notebook e ti consentirà di lavorare più facilmente con Neptune ML.

Per istruzioni su come configurare i notebook in SageMaker, consulta Avvio del progetto graph-notebook con Amazon SageMaker.

Per istruzioni su come installare e configurare Neptune stesso, consulta Configurazione di Neptune.

È inoltre possibile collegare un'installazione locale dei notebook Neptune a un cluster database Neptune. Questa operazione è leggermente più complessa perché i cluster database Amazon Neptune possono essere creati solo in un Amazon Virtual Private Cloud (Amazon VPC) che, per impostazione predefinita, è isolato dal mondo esterno. Esistono diversi modi per connettersi a un VPC dall'esterno. Uno consiste nell'utilizzare un sistema di bilanciamento del carico. Un altro metodo prevede l'uso del peering VPC (consulta la guida Amazon Virtual Private Cloud Peering).

La soluzione più pratica per la maggior parte delle persone, tuttavia, è connettersi per configurare un server proxy Amazon EC2 all'interno del VPC e quindi utilizzare il tunnelling SSH (denominato anche port fowarding) per connettersi ad esso. Puoi trovare le istruzioni per la configurazione in Connecting graph notebook localmente ad Amazon Neptune nella cartella `additional-databases/neptune` del progetto GitHub graph-notebook.

Utilizzo dei notebook Neptune con database a grafo open source

Per iniziare a utilizzare gratuitamente la tecnologia dei grafi, puoi anche usare i notebook Neptune con vari database open source sul back-end. Alcuni esempi includono il server Gremlin TinkerPop e il database Blazegraph.

Per utilizzare Gremlin Server come database di back-end, segui le istruzioni su:

- Cartella Connecting graph-notebook to a Gremlin Server in GitHub.
- Cartella graph-notebook Gremlin configuration in GitHub.

Per utilizzare un'istanza locale di Blazegraph come database back-end, segui queste istruzioni:

- Istruzioni per l'avvio rapido di Blazegraph
- Cartella graph-notebook Blazegraph configuration in GitHub.

Migrazione dei notebook Neptune da Jupyter a JupyterLab 3

I notebook Neptune creati prima del 21 dicembre 2022 utilizzano l'ambiente Amazon Linux 1. Puoi eseguire la migrazione dei notebook Jupyter precedenti creati prima di quella data al nuovo ambiente Amazon Linux 2 con JupyterLab 3 seguendo la procedura descritta in questo post del blog AWS: Migrate your work to an Amazon SageMaker notebook instance with Amazon Linux 2.

Inoltre, esistono alcuni passaggi aggiuntivi che si applicano specificamente alla migrazione dei notebook Neptune al nuovo ambiente:

Prerequisiti specifici di Neptune

Nel ruolo IAM del notebook Neptune di origine, aggiungi tutte le autorizzazioni seguenti:

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:ListBucket",
    "s3:CreateBucket",
    "s3:PutObject"
  ],
  "Resource": [
    "arn:aws:s3:::(your ebs backup bucket name)",
    "arn:aws:s3:::(your ebs backup bucket name)/*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "sagemaker:ListTags"
  ],
  "Resource": [
    "*"
  ]
}
```

Assicurati di specificare il nome ARN corretto per il bucket S3 che verrà usato per il backup.

Configurazione del ciclo di vita specifico di Neptune

Quando si crea il secondo script di configurazione del ciclo di vita per il ripristino del backup (da `on-create.sh`) come descritto nel post del blog, il nome del ciclo di vita deve seguire il formato `aws-neptune-*`, ad esempio `aws-neptune-sync-from-s3` Questo garantisce che sia possibile selezionare la configurazione del ciclo di vita durante la creazione del notebook nella console Neptune.

Sincronizzazione specifica di Neptune da uno snapshot a una nuova istanza

Per i passaggi descritti nel post del blog per la sincronizzazione da uno snapshot a una nuova istanza, ecco le modifiche specifiche di Neptune:

- Nel passaggio 4 scegli notebook-al2-v2.
- Nel passaggio 5 riutilizza il ruolo IAM del notebook Neptune di origine.
- Tra i passaggi 7 e 8:
  - In Impostazioni dell'istanza notebook imposta un nome con il formato `aws-neptune-*`.
  - Apri il gruppo di elementi comprimibili (accordion) delle impostazioni di Rete e per VPC, sottorete e gruppo di sicurezza seleziona gli stessi valori del notebook di origine.

Passaggi specifici di Neptune dopo la creazione del nuovo notebook

1. Seleziona il pulsante Apri Jupyter per il notebook. Una volta visualizzato il file `SYNC_COMPLETE` nella directory principale, procedi al passaggio successivo.
2. Passa alla pagina dell'istanza notebook nella console SageMaker.
3. Arresta il notebook.
4. Seleziona Edit (Modifica).
5. Nelle impostazioni dell'istanza notebook modifica il campo Configurazione del ciclo di vita selezionando il ciclo di vita originale del notebook Neptune di origine. Tieni presente che questo non è il ciclo di vita del backup EBS.
6. Seleziona Aggiorna impostazioni notebook.
7. Riavvia il notebook.

Con le modifiche ai passaggi illustrati nel post del blog, i notebook per grafi dovrebbero ora essere stati trasferiti in una nuova istanza notebook Neptune che utilizza l'ambiente Amazon Linux 2 e JupyterLab 3. Verranno visualizzati per l'accesso e la gestione nella pagina di Neptune nella AWS Management Console e puoi ora riprendere da dove avevi interrotto selezionando Apri Jupyter o Apri JupyterLab.

Utilizzo dei comandi magic di Neptune Workbench nei notebook

Neptune Workbench offre numerosi comandi magic per i notebook che ti consentono di risparmiare tempo ed energie. Si dividono in due categorie: comandi magic di riga e comandi magic di cella.

I comandi magic di riga sono comandi preceduti da un unico segno di percentuale (%). Accettano solo l'intera riga come input, non l'input dal resto del corpo della cella. Neptune Workbench offre i comandi magic di riga seguenti:

- %seed
- %load
- %load_ids
- %load_status
- %cancel_load
- %status
- %gremlin_status
- %opencypher_status o %oc_status
- %stream_viewer
- %sparql_status
- %graph_notebook_config
- %graph_notebook_host
- %graph_notebook_version
- %graph_notebook_vis_options
- %statistics
- %summary

I comandi magic di cella sono preceduti da due segni di percentuale (%%) anziché uno e utilizzano il contenuto della cella come input, sebbene possano anche accettare come input il contenuto della riga. Neptune Workbench offre i comandi magic di cella seguenti:

- %%sparql
- %%gremlin
- %%opencypher o %%oc
- %%graph_notebook_config
- %%graph_notebook_vis_options

Esistono anche due comandi magic, un comando magic di riga e un comando magic di cella, per Neptune Machine Learning:

- %neptune_ml
- %%neptune_ml

> **Note**
>
> Quando si usano i comandi magic di Neptune, in genere è possibile visualizzare il testo della guida utilizzando un parametro `--help` o `-h`. Con un comando magic di cella, il corpo non può essere vuoto, quindi per visualizzare la guida è necessario immettere un testo di riempimento, anche un singolo carattere, nel corpo. Per esempio:
>
> ```
> %%gremlin --help
>   x
> ```

Inserimento di variabili nei comandi magic di riga o di cella

È possibile fare riferimento alle variabili definite in un notebook all'interno di qualsiasi comando magic di cella o di riga nel notebook utilizzando il formato: `${VAR_NAME}`.

Si supponga ad esempio che tu definisca queste variabili:

```
c = 'code'
my_edge_labels = '{"route":"dist"}'
```

Quindi, questa query Gremlin in un comando magic di cella:

```
%%gremlin -de $my_edge_labels
g.V().has('${c}','SAF').out('route').values('${c}')
```

Questo equivale a:

```
%%gremlin -de {"route":"dist"}
g.V().has('code','SAF').out('route').values('code')
```

Argomenti di query che funzionano con tutti i linguaggi di query

I seguenti argomenti di query funzionano con i comandi magic `%%gremlin`, `%%opencypher` e `%%sparql` in Neptune Workbench:

Argomenti di query comuni

- **`--store-to`** (o **`-s`**): specifica il nome di una variabile in cui archiviare i risultati della query.
- **`--silent`**: se presente, non viene visualizzato alcun output dopo il completamento della query.

- **`--group-by`** (o **`-g`**): specifica la proprietà utilizzata per raggruppare i nodi (ad esempio `code` o `T.region`). I vertici sono colorati in base al gruppo assegnato.
- **`--ignore-groups`**: se presente, tutte le opzioni di raggruppamento vengono ignorate.
- **`--display-property`** (o **`-d`**): specifica la proprietà di cui occorre visualizzare il valore per ogni vertice.

  Il valore predefinito per ogni linguaggio di query è il seguente:

  - Per Gremlin: `T.label`.
  - Per openCypher: `~labels`.
  - Per SPARQL: `type`.
- **`--edge-display-property`** (o **`-t`**): specifica la proprietà di cui occorre visualizzare il valore per ogni arco.

  Il valore predefinito per ogni linguaggio di query è il seguente:

  - Per Gremlin: `T.label`.
  - Per openCypher: `~labels`.
  - Per SPARQL: `type`.
- **`--tooltip-property`** (o **`-de`**): specifica una proprietà di cui occorre visualizzare il valore come descrizione comando per ogni nodo.

  Il valore predefinito per ogni linguaggio di query è il seguente:

  - Per Gremlin: `T.label`.
  - Per openCypher: `~labels`.
  - Per SPARQL: `type`.
- **`--edge-tooltip-property`** (o **`-te`**): specifica una proprietà di cui occorre visualizzare il valore come descrizione comando per ogni arco.

  Il valore predefinito per ogni linguaggio di query è il seguente:

  - Per Gremlin: `T.label`.
  - Per openCypher: `~labels`.

  - Per SPARQL: `type`.
- **`--label-max-length`** (o **`-l`**): specifica la lunghezza di caratteri massima di tutte le etichette dei vertici. Il valore predefinito è 10.
- **`--edge-label-max-length`** (o **`-le`**): specifica la lunghezza di caratteri massima di tutte le etichette degli archi. Il valore predefinito è 10.

  Solo nel caso di openCypher, è `--rel-label-max-length` o `-rel`.
- **`--simulation-duration`** (o **`-sd`**): specifica la durata massima della simulazione delle proprietà fisiche della visualizzazione. Il valore predefinito è 1.500 ms.
- **`--stop-physics`** (o **`-sp`**): disabilita le proprietà fisiche della visualizzazione dopo che la simulazione iniziale si è stabilizzata.

I valori delle proprietà per questi argomenti possono essere costituiti da una singola chiave della proprietà o da una stringa JSON che può specificare una proprietà diversa per ogni tipo di etichetta. È possibile specificare una stringa JSON solo usando l'inserimento di variabili.

Comando magic di riga **`%seed`**

Il comando magic di riga `%seed` è un modo pratico per aggiungere dati all'endpoint Neptune che puoi usare per esplorare e sperimentare con le query Gremlin, openCypher o SPARQL. Fornisce un modulo in cui è possibile selezionare il modello di dati che si desidera esplorare (property-graph o RDF) e quindi scegliere tra numerosi set di dati di esempio diversi forniti da Neptune.

Comando magic di riga **`%load`**

Il comando magic di riga `%load` genera un modulo che puoi utilizzare per inviare una richiesta di caricamento in blocco a Neptune (consulta Comando dello strumento di caricamento Neptune). L'origine deve essere un percorso Amazon S3 nella stessa regione del cluster Neptune.

Comando magic di riga **`%load_ids`**

Il comando magic di riga `%load_ids` recupera gli ID di caricamento che sono stati inviati all'endpoint host del notebook (consulta Parametri della richiesta Get-Status dello strumento di caricamento Neptune). Il formato della richiesta è il seguente:

```
GET https://your-neptune-endpoint:port/loader
```

Comando magic di riga **`%load_status`**

Il comando magic di riga `%load_status` recupera lo stato di caricamento di un processo di caricamento specifico che è stato inviato all'endpoint host del notebook, specificato dall'input di riga (consulta Parametri della richiesta Get-Status dello strumento di caricamento Neptune). Il formato della richiesta è il seguente:

```
GET https://your-neptune-endpoint:port/loader?loadId=loadId
```

Il comando magic di riga ha l'aspetto seguente:

```
%load_status load id
```

Comando magic di riga **`%cancel_load`**

Il comando magic di riga `%cancel_load` annulla un processo di caricamento specifico (consulta Annullamento di un processo dello strumento di caricamento Neptune). Il formato della richiesta è il seguente:

```
DELETE https://your-neptune-endpoint:port/loader?loadId=loadId
```

Il comando magic di riga ha l'aspetto seguente:

```
%cancel_load load id
```

Comando magic di riga **`%status`**

Recupera le informazioni sullo stato dall'endpoint host del notebook (%graph_notebook_config mostra l'endpoint host).

Comando magic di riga **`%gremlin_status`**

Recupera le informazioni sullo stato delle query Gremlin.

Comando magic di riga **`%opencypher_status`** (anche **`%oc_status`**)

Recupera lo stato della query per una query openCypher. Questo comando magic di riga accetta gli argomenti facoltativi seguenti:

- **`--queryId`** o **`-q`**: specifica l'ID di una query in esecuzione specifica per cui mostrare lo stato.
- **`--cancel_query`** o **`-c`**: annulla una query in esecuzione. Non accetta un valore.
- **`--silent`** o **`-s`**: se l'argomento `--silent` è impostato su `true` quando si annulla una query, la query in esecuzione viene annullata con un codice di risposta HTTP `200`. Altrimenti, il codice di risposta HTTP è `500`.
- **`--store-to`**: specifica il nome di una variabile in cui archiviare i risultati della query.

Comando magic di riga **`%sparql_status`**

Recupera le informazioni sullo stato delle query SPARQL.

Comando magic di riga **`%stream_viewer`**

Il comando magic di riga `%stream_viewer` mostra un'interfaccia che consente di esplorare in modo interattivo le voci registrate nei flussi Neptune, se i flussi sono abilitati sul cluster Neptune. Accetta gli argomenti facoltativi seguenti:

- **`language`**: linguaggio di query dei dati del flusso: `gremlin` o `sparql`. L'impostazione predefinita, se non si specifica questo argomento, è `gremlin`.
- **`--limit`**: specifica il numero massimo di voci del flusso da visualizzare per pagina. Il valore predefinito, se non si specifica questo argomento, è `10`.

> **Note**
>
> Il comando magic di riga `%stream_viewer` è completamente supportato solo nelle versioni del motore 1.0.5.1 e precedenti.

Comando magic di riga **`%graph_notebook_config`**

Questo comando magic di riga mostra un oggetto JSON contenente la configurazione utilizzata dal notebook per comunicare con Neptune. La configurazione include:

- `host`: endpoint a cui connettersi e inviare i comandi.
- `port`: porta utilizzata per inviare comandi a Neptune. Il valore predefinito è `8182`.

- `auth_mode`: modalità di autenticazione da utilizzare per l'invio dei comandi a Neptune. Deve essere `IAM` se ci si connette a un cluster per cui è abilitata l'autenticazione IAM; in caso contrario `DEFAULT`.
- `load_from_s3_arn`: specifica un nome ARN Amazon S3 per il comando magic `%load` da usare. Se questo valore è vuoto, il nome ARN deve essere specificato nel comando `%load`.
- `ssl`: valore booleano che indica se connettersi o meno a Neptune tramite TLS. Il valore predefinito è `true`.
- `aws_region`: regione in cui è implementato questo notebook. Queste informazioni vengono utilizzate per l'autenticazione IAM e per le richieste `%load`.

Per modificare la configurazione, copia l'output di `%graph_notebook_config` in una nuova cella e introduci le modifiche desiderate. Se quindi esegui il comando magic di cella [%%graph_notebook_config] nella nuova cella, la configurazione verrà modificata di conseguenza.

Comando magic di riga **`%graph_notebook_host`**

Imposta l'input di riga come host del notebook.

Comando magic di riga **`%graph_notebook_version`**

Il comando magic di riga `%graph_notebook_version` restituisce il numero di rilascio del notebook Neptune Workbench. Ad esempio, la visualizzazione dei grafi è stata introdotta nella versione `1.27`.

Comando magic di riga **`%graph_notebook_vis_options`**

Il comando magic di riga `%graph_notebook_vis_options` mostra le impostazioni di visualizzazione correnti utilizzate dal notebook. Queste opzioni sono spiegate nella documentazione di vis.js.

Per modificare queste impostazioni, copia l'output in una nuova cella, apporta le modifiche desiderate e quindi esegui il comando magic di cella `%%graph_notebook_vis_options` sulla cella.

Per ripristinare le impostazioni di visualizzazione predefinite, puoi eseguire il comando magic di riga `%graph_notebook_vis_options` con un parametro `reset`. Tutte le impostazioni di visualizzazione verranno ripristinate:

```
%graph_notebook_vis_options reset
```

Comando magic di riga **`%statistics`**

Il comando magic di riga `%statistics` consente di recuperare o gestire le statistiche del motore DFE (consulta Gestione delle statistiche utilizzabili dal motore DFE Neptune). Questo comando magic consente anche di recuperare un riepilogo del grafo.

Accetta i seguenti parametri:

- **`--language`**: linguaggio di query dell'endpoint delle statistiche: o `propertygraph` (o pg) o `rdf`.

  Se non viene specificato, il valore predefinito è `propertygraph`.
- **`--mode`** (o **`-m`**): specifica il tipo di richiesta o azione da inviare: una tra le seguenti: `status`, `disableAutoCompute`, `enableAutoCompute`,`refresh`, `delete`, `detailed` o `basic`).

  Se non viene specificato, il valore predefinito è `status` a meno che non sia specificato `--summary`, nel qual caso l'impostazione predefinita è `basic`.
- **`--summary`**: recupera il riepilogo del grafo dall'endpoint di riepilogo delle statistiche del linguaggio selezionato.
- **`--silent`**: se presente, non viene visualizzato alcun output dopo il completamento della query.
- **`--store-to`**: consente di specificare una variabile in cui archiviare i risultati della query.

Comando magic di riga **`%summary`**

Il comando magic di riga `%summary` consente di recuperare le informazioni di riepilogo del grafo. È disponibile a partire dalla versione del motore Neptune `1.2.1.0`.

Accetta i seguenti parametri:

- **`--language`**: linguaggio di query dell'endpoint delle statistiche: o `propertygraph` (o pg) o `rdf`.

  Se non viene specificato, il valore predefinito è `propertygraph`.
- **`--detailed`**: attiva o disattiva la visualizzazione dei campi delle strutture nell'output.

  Se non viene specificato, l'impostazione predefinita è la modalità di visualizzazione del riepilogo `basic`.
- **`--silent`**: se presente, non viene visualizzato alcun output dopo il completamento della query.
- **`--store-to`**: consente di specificare una variabile in cui archiviare i risultati della query.

Comando magic di cella **%%graph_notebook_config**

Il comando magic di cella `%%graph_notebook_config` utilizza un oggetto JSON contenente le informazioni di configurazione per modificare le impostazioni utilizzate dal notebook per comunicare con Neptune, se possibile. La configurazione accetta lo stesso formato restituito dal comando magic di riga %graph_notebook_config.

Per esempio:

```
%%graph_notebook_config
{
  "host": "my-new-cluster-endpoint.amazon.com",
  "port": 8182,
  "auth_mode": "DEFAULT",
  "load_from_s3_arn": "",
  "ssl": true,
  "aws_region": "us-east-1"
}
```

Comando magic di cella **%%sparql**

Il comando magic di cella `%%sparql` invia una query SPARQL all'endpoint Neptune. Accetta gli input di riga facoltativi seguenti:

- **`-h`** o **`--help`**: restituisce il testo della guida relativo a questi parametri.
- **`--path`**: aggiunge un prefisso al percorso dell'endpoint SPARQL. Ad esempio, se si specifica `--path "abc/def"`, l'endpoint chiamato sarà *`host`*`:`*`port`*`/abc/def`.
- **`--expand-all`**: si tratta di suggerimento per la visualizzazione delle query che indica al visualizzatore di includere tutti i risultati `?s ?p ?o` nel diagramma del grafo, indipendentemente dal tipo di associazione.

  Per impostazione predefinita, una visualizzazione SPARQL include solo modelli di triple in cui `o?` è un `uri` o un `bnode` (nodo vuoto). Tutti gli altri tipi di `?o` associazione, come stringhe letterali o numeri interi, vengono trattati come proprietà del nodo `?s` che possono essere visualizzate nel riquadro Dettagli nella scheda Grafo.

  Utilizza invece l'hint di query `--expand-all` quando desideri includere valori letterali come i vertici nella visualizzazione.

Non combinare questo suggerimento di visualizzazione con i parametri di spiegazione, in quanto le query EXPLAIN non vengono visualizzate.

- **`--explain-type`**: consente di specificare la modalità di spiegazione da utilizzare (tra le seguenti: `dynamic`, `static` o `details`).
- **`--explain-format`**: consente di specificare il formato della risposta per una query di spiegazione (`text/csv` o `text/html`).
- --store-to: consente di specificare una variabile in cui archiviare i risultati della query.

Esempio di query `explain`:

```
%%sparql explain

SELECT * WHERE {?s ?p ?o} LIMIT 10
```

Esempio di una query di visualizzazione con un parametro di suggerimento di visualizzazione `--expand-all`(consulta Visualizzazione SPARQL):

```
%%sparql --expand-all

SELECT * WHERE {?s ?p ?o} LIMIT 10
```

Comando magic di cella **`%%gremlin`**

Il comando magic di cella `%%gremlin` invia una query Gremlin all'endpoint Neptune tramite WebSocket. Accetta un input di riga facoltativo per passare alla modalità Gremlin `explain` /> o API Gremlin `profile` e un input suggerimento di visualizzazione facoltativo separato per modificare il comportamento dell'output della visualizzazione (consulta Visualizzazione Gremlin).

Esempio di query `explain`:

```
%%gremlin explain

g.V().limit(10)
```

Esempio di query `profile`:

```
%%gremlin profile
```

```
g.V().limit(10)
```

Esempio di una query di visualizzazione con un hint di query di visualizzazione:

```
%%gremlin -p v,outv

g.V().out().limit(10)
```

Parametri facoltativi per le query **`%%gremlin profile`**

- **`--chop`**: specifica la lunghezza massima della stringa dei risultati del profilo. Il valore predefinito, se non si specifica questo argomento, è 250.
- **`--serializer`**: specifica il serializzatore da utilizzare per i risultati. I valori consentiti sono tutti i valori di enumerazione "Serializers" di tipo MIME o del driver TinkerPop validi. Il valore predefinito, se non si specifica questo argomento, è `application.json`.
- **`--no-results`**: visualizza solo il numero di risultati. Se non viene utilizzato, per impostazione predefinita nel report del profilo vengono visualizzati tutti i risultati della query.
- **`--indexOps`**: mostra un report dettagliato di tutte le operazioni sull'indice.

Comando magic di cella **`%%opencypher`** (anche **`%%oc`**)

Il comando magic di cella `%%opencypher` (che ha anche la forma abbreviata `%%oc`) invia una query openCypher all'endpoint Neptune. Accetta gli argomenti di input di riga facoltativi seguenti:

- mode: modalità di query: `query` o `bolt`. Il valore predefinito, se non si specifica questo argomento, è `query`.
- **`--group-by`** o **`-g`**: specifica la proprietà utilizzata per raggruppare i nodi. Ad esempio, `code, ~id`. Il valore predefinito, se non si specifica questo argomento, è `~labels`.
- **`--ignore-groups`**: se presente, tutte le opzioni di raggruppamento vengono ignorate.
- **`--display-propery`** o **`-d`**: specifica la proprietà di cui occorre visualizzare il valore per ogni vertice. Il valore predefinito, se non si specifica questo argomento, è `~labels`.
- **`--edge-display-propery`** o **`-de`**: specifica la proprietà di cui occorre visualizzare il valore per ogni arco. Il valore predefinito, se non si specifica questo argomento, è `~labels`.
- **`--label-max-length`** o **`-l`**: specifica il numero massimo di caratteri di un'etichetta di vertice da visualizzare. Il valore predefinito, se non si specifica questo argomento, è `10`.

- **`--store-to`** o **`-s`**: specifica il nome di una variabile in cui archiviare i risultati della query.
- **`--plan-cache`** o **`-pc`**: specifica la modalità cache del piano da utilizzare. Il valore predefinito è `auto`.
- **`--query-timeout`** o **`-qt`**: specifica il timeout di query massimo in millisecondi. Il valore predefinito è `1800000`.
- **`--query-parameters`** o **`qp`**: definizioni dei parametri da applicare alla query. Questa opzione può accettare il nome di una singola variabile o una rappresentazione in formato stringa della mappa.

  Esempio di utilizzo di **`--query-parameters`**

  1. Definisci una mappa dei parametri OpenCypher in una cella del notebook.

     ```
     params = '''{
      "name":"john",
      "age": 20,
     }'''
     ```

  2. Passa i parametri in `--query-parameters` in un'altra cella con `%%oc`.

     ```
     %%oc --query-parameters params

     MATCH (n {name: $name, age: $age})
     RETURN n
     ```

Comando magic di cella **`%%graph_notebook_vis_options`**

Il comando magic di cella `%%graph_notebook_vis_options` consente di impostare le opzioni di visualizzazione per il notebook. È possibile copiare le impostazioni restituite dal comando magic di riga `%graph-notebook-vis-options` in una nuova cella, modificarle e utilizzare il comando magic di cella `%%graph_notebook_vis_options` per impostare i nuovi valori.

Queste opzioni sono spiegate nella documentazione di vis.js.

Per ripristinare le impostazioni di visualizzazione predefinite, puoi eseguire il comando magic di riga `%graph_notebook_vis_options` con un parametro `reset`. Tutte le impostazioni di visualizzazione verranno ripristinate:

```
%graph_notebook_vis_options reset
```

Comando magic di riga **`%neptune_ml`**

Puoi usare il comando magic di riga `%neptune_ml` per avviare e gestire varie operazioni di Neptune ML.

> **Note**
>
> Puoi anche avviare e gestire alcune operazioni di Neptune ML usando il comando magic di cella %%neptune_ml.

- **`%neptune_ml export start`**: avvia un nuovo processo di esportazione.

  Parameters (Parametri)

  - **`--export-url`** *`exporter-endpoint`* (facoltativo): endpoint Gateway Amazon API in cui è possibile chiamare l'esportatore.
  - **`--export-iam`** (facoltativo): flag che indica che le richieste all'URL di esportazione devono essere firmate tramite SigV4.
  - **`--export-no-ssl`** (facoltativo): flag che indica che il protocollo SSL non deve essere utilizzato durante la connessione all'esportatore.
  - **`--wait`** (facoltativo): flag che indica che l'operazione deve attendere il completamento dell'esportazione.
  - **`--wait-interval`** *`interval-to-wait`* (facoltativo): imposta il tempo, in secondi, tra i controlli dello stato dell'esportazione (valore predefinito: 60).
  - **`--wait-timeout`** *`timeout-seconds`* (facoltativo): imposta il tempo, in secondi, di attesa del completamento del processo di esportazione prima di restituire lo stato più recente (impostazione predefinita: 3.600).
  - **`--store-to`** *`location-to-store-result`* (facoltativo): variabile in cui archiviare il risultato dell'esportazione. Se è specificato il flag `--wait`, lo stato finale verrà archiviato in tale posizione.
- **`%neptune_ml export status`**: recupera lo stato di un processo di esportazione.

Parameters (Parametri)

- **`--job-id`** *`ID processo di esportazione`*: ID del processo di esportazione di cui recuperare lo stato.
- **`--export-url`** *`exporter-endpoint`* (facoltativo): endpoint Gateway Amazon API in cui è possibile chiamare l'esportatore.
- **`--export-iam`** (facoltativo): flag che indica che le richieste all'URL di esportazione devono essere firmate tramite SigV4.
- **`--export-no-ssl`** (facoltativo): flag che indica che il protocollo SSL non deve essere utilizzato durante la connessione all'esportatore.
- **`--wait`** (facoltativo): flag che indica che l'operazione deve attendere il completamento dell'esportazione.
- **`--wait-interval`** *`interval-to-wait`* (facoltativo): imposta il tempo, in secondi, tra i controlli dello stato dell'esportazione (valore predefinito: 60).
- **`--wait-timeout`** *`timeout-seconds`* (facoltativo): imposta il tempo, in secondi, di attesa del completamento del processo di esportazione prima di restituire lo stato più recente (impostazione predefinita: 3.600).
- **`--store-to`** *`location-to-store-result`* (facoltativo): variabile in cui archiviare il risultato dell'esportazione. Se è specificato il flag `--wait`, lo stato finale verrà archiviato in tale posizione.

- **`%neptune_ml dataprocessing start`**: avvia la fase di elaborazione dei dati di Neptune ML.

  Parameters (Parametri)

  - **`--job-id`** *`ID per questo processo`* (facoltativo): ID da assegnare a questo processo.
  - **`--s3-input-uri`** *`URI S3`* (facoltativo): URI S3 in cui trovare l'input per questo processo di elaborazione dati.
  - **`--config-file-name`** *`nome file`* (facoltativo): nome del file di configurazione per questo processo di elaborazione dati.
  - **`--store-to`** *`location-to-store-result`* (facoltativo): variabile in cui archiviare il risultato dell'elaborazione dati.
  - **`--instance-type`** *`(tipo di istanza)`* (facoltativo): dimensioni dell'istanza da utilizzare per questo processo di elaborazione dati.
  - **`--wait`** (facoltativo): flag che indica che l'operazione deve attendere il completamento dell'elaborazione dati.

  - **`--wait-interval`** *`interval-to-wait`* (facoltativo): imposta il tempo, in secondi, tra i controlli dello stato dell'elaborazione dati (valore predefinito: 60).
  - **`--wait-timeout`** *`timeout-seconds`* (facoltativo): imposta il tempo, in secondi, di attesa del completamento del processo di elaborazione dati prima di restituire lo stato più recente (impostazione predefinita: 3.600).
- **`%neptune_ml dataprocessing status`**: recupera lo stato di un processo di elaborazione dati.

  Parameters (Parametri)

  - **`--job-id`** *`ID processo`*: ID del processo di cui recuperare lo stato.
  - **`--store-to`** *`tipo di istanza`* (facoltativo): variabile in cui archiviare il risultato dell'addestramento del modello.
  - **`--wait`** (facoltativo): flag che indica che l'operazione deve attendere il completamento dell'addestramento del modello.
  - **`--wait-interval`** *`interval-to-wait`* (facoltativo): imposta il tempo, in secondi, tra i controlli dello stato dell'addestramento del modello (valore predefinito: 60).
  - **`--wait-timeout`** *`timeout-seconds`* (facoltativo): imposta il tempo, in secondi, di attesa del completamento del processo di elaborazione dati prima di restituire lo stato più recente (impostazione predefinita: 3.600).
- **`%neptune_ml training start`**: avvia il processo di addestramento del modello Neptune ML.

  Parameters (Parametri)

  - **`--job-id`** *`ID per questo processo`* (facoltativo): ID da assegnare a questo processo.
  - **`--data-processing-id`***`ID del processo di elaborazione dati`* (facoltativo): ID del processo di elaborazione dati che ha creato gli artefatti da utilizzare per l'addestramento.
  - **`--s3-output-uri`** *`URI S3`* (facoltativo): URI S3 in cui archiviare l'output di questo processo di addestramento del modello.
  - **`--instance-type`** *`(tipo di istanza)`* (facoltativo): dimensioni dell'istanza da utilizzare per questo processo di addestramento del modello.
  - **`--store-to`** *`location-to-store-result`* (facoltativo): variabile in cui archiviare il risultato dell'addestramento del modello.
  - **`--wait`** (facoltativo): flag che indica che l'operazione deve attendere il completamento dell'addestramento del modello.

  - **`--wait-interval`** *`interval-to-wait`* (facoltativo): imposta il tempo, in secondi, tra i controlli dello stato dell'addestramento del modello (valore predefinito: 60).
  - **`--wait-timeout`** *`timeout-seconds`* (facoltativo): imposta il tempo, in secondi, di attesa del completamento del processo di addestramento del modello prima di restituire lo stato più recente (impostazione predefinita: 3.600).
- **`%neptune_ml training status`**: recupera lo stato di un processo di addestramento del modello Neptune ML.

  Parameters (Parametri)

  - **`--job-id`** *`ID processo`*: ID del processo di cui recuperare lo stato.
  - **`--store-to`** *`tipo di istanza`* (facoltativo): variabile in cui archiviare il risultato dello stato.
  - **`--wait`** (facoltativo): flag che indica che l'operazione deve attendere il completamento dell'addestramento del modello.
  - **`--wait-interval`** *`interval-to-wait`* (facoltativo): imposta il tempo, in secondi, tra i controlli dello stato dell'addestramento del modello (valore predefinito: 60).
  - **`--wait-timeout`** *`timeout-seconds`* (facoltativo): imposta il tempo, in secondi, di attesa del completamento del processo di elaborazione dati prima di restituire lo stato più recente (impostazione predefinita: 3.600).
- **`%neptune_ml endpoint create`**: crea un endpoint di query per un modello Neptune ML.

  Parameters (Parametri)

  - **`--job-id`** *`ID per questo processo`* (facoltativo): ID da assegnare a questo processo.
  - **`--model-job-id`** *`ID processo di addestramento del modello`* (facoltativo): ID del processo di addestramento del modello per il quale creare un endpoint di query.
  - **`--instance-type`** *`(tipo di istanza)`* (facoltativo): dimensioni dell'istanza da utilizzare per l'endpoint di query.
  - **`--store-to`** *`location-to-store-result`* (facoltativo): variabile in cui archiviare il risultato della creazione dell'endpoint.
  - **`--wait`** (facoltativo): flag che indica che l'operazione deve attendere il completamento della creazione dell'endpoint.
  - **`--wait-interval`** *`interval-to-wait`* (facoltativo): imposta il tempo, in secondi, tra i controlli dello stato (valore predefinito: 60).

- **`--wait-timeout`** *`timeout-seconds`* (facoltativo): imposta il tempo, in secondi, di attesa del completamento del processo di creazione dell'endpoint prima di restituire lo stato più recente (impostazione predefinita: 3.600).

- **`%neptune_ml endpoint status`**: recupera lo stato di un endpoint di query Neptune ML.

  Parameters (Parametri)

  - **`--job-id`** *`ID di creazione dell'endpoint`* (facoltativo): ID di un processo di creazione endpoint per cui segnalare lo stato.
  - **`--store-to`** *`location-to-store-result`* (facoltativo): variabile in cui archiviare il risultato dello stato.
  - **`--wait`** (facoltativo): flag che indica che l'operazione deve attendere il completamento della creazione dell'endpoint.
  - **`--wait-interval`** *`interval-to-wait`* (facoltativo): imposta il tempo, in secondi, tra i controlli dello stato (valore predefinito: 60).
  - **`--wait-timeout`** *`timeout-seconds`* (facoltativo): imposta il tempo, in secondi, di attesa del completamento del processo di creazione dell'endpoint prima di restituire lo stato più recente (impostazione predefinita: 3.600).

Comando magic di cella **`%%neptune_ml`**

Il comando magic di cella `%%neptune_ml` ignora gli input di riga come `--job-id` o `--export-url`. Invece, consente di fornire tali input e altri all'interno del corpo della cella.

È inoltre possibile salvare questi input in un'altra cella, assegnata a una variabile Jupyter, e quindi inserirli nel corpo della cella usando tale variabile. In questo modo, è possibile usare tali input più e più volte senza doverli reinserire tutti ogni volta.

Funziona solo se la variabile di inserimento è l'unico contenuto della cella. Non è possibile utilizzare più variabili in una cella o una combinazione di testo e variabile.

Ad esempio, il comando magic di cella `%%neptune_ml export start` può utilizzare un documento JSON nel corpo della cella che contiene tutti i parametri descritti in Parametri utilizzati per controllare il processo di esportazione di Neptune.

Nel notebook Neptune-ML-01-Introduction-to-Node-Classification-Gremlin, in Configurazione delle funzionalità nella sezione Esportazione della configurazione dei dati e del modello puoi osservare

che la seguente cella contiene i parametri di esportazione in un documento assegnato a una variabile Jupyter denominata `export-params`:

```
export_params = {
  "command": "export-pg",
  "params": {
    "endpoint": neptune_ml.get_host(),
    "profile": "neptune_ml",
    "useIamAuth": neptune_ml.get_iam(),
    "cloneCluster": False
   },
  "outputS3Path": f'{s3_bucket_uri}/neptune-export',
  "additionalParams": {
    "neptune_ml": {
      "targets": [
        {
          "node": "movie",
          "property": "genre"
        }
      ],
      "features": [
        {
          "node": "movie",
          "property": "title",
          "type": "word2vec"
        },
        {
          "node": "user",
          "property": "age",
          "type": "bucket_numerical",
          "range" : [1, 100],
          "num_buckets": 10
        }
      ]
    }
  },
  "jobSize": "medium"}
```

Quando esegui questa cella, Jupyter salva il documento dei parametri con tale nome. Quindi, puoi usare `${export_params}` per inserire il documento JSON nel corpo di una cella `%%neptune_ml export start cell`, nel modo seguente:

```
%%neptune_ml export start --export-url {neptune_ml.get_export_service_host()} --export-
iam --wait --store-to export_results

${export_params}
```

Formati disponibili del comando magic di cella **%%neptune_ml**

Il comando magic di cella `%%neptune_ml` può essere usato nei formati seguenti:

- **`%%neptune_ml export start`**: avvia un processo di esportazione Neptune ML.
- **`%%neptune_ml dataprocessing start`**: avvia un processo di elaborazione dati Neptune ML.
- **`%%neptune_ml training start`**: avvia un processo di addestramento del modello Neptune ML.
- **`%%neptune_ml endpoint create`**: crea un endpoint di query Neptune ML per un modello.

Visualizzazione dei grafi in Neptune Workbench

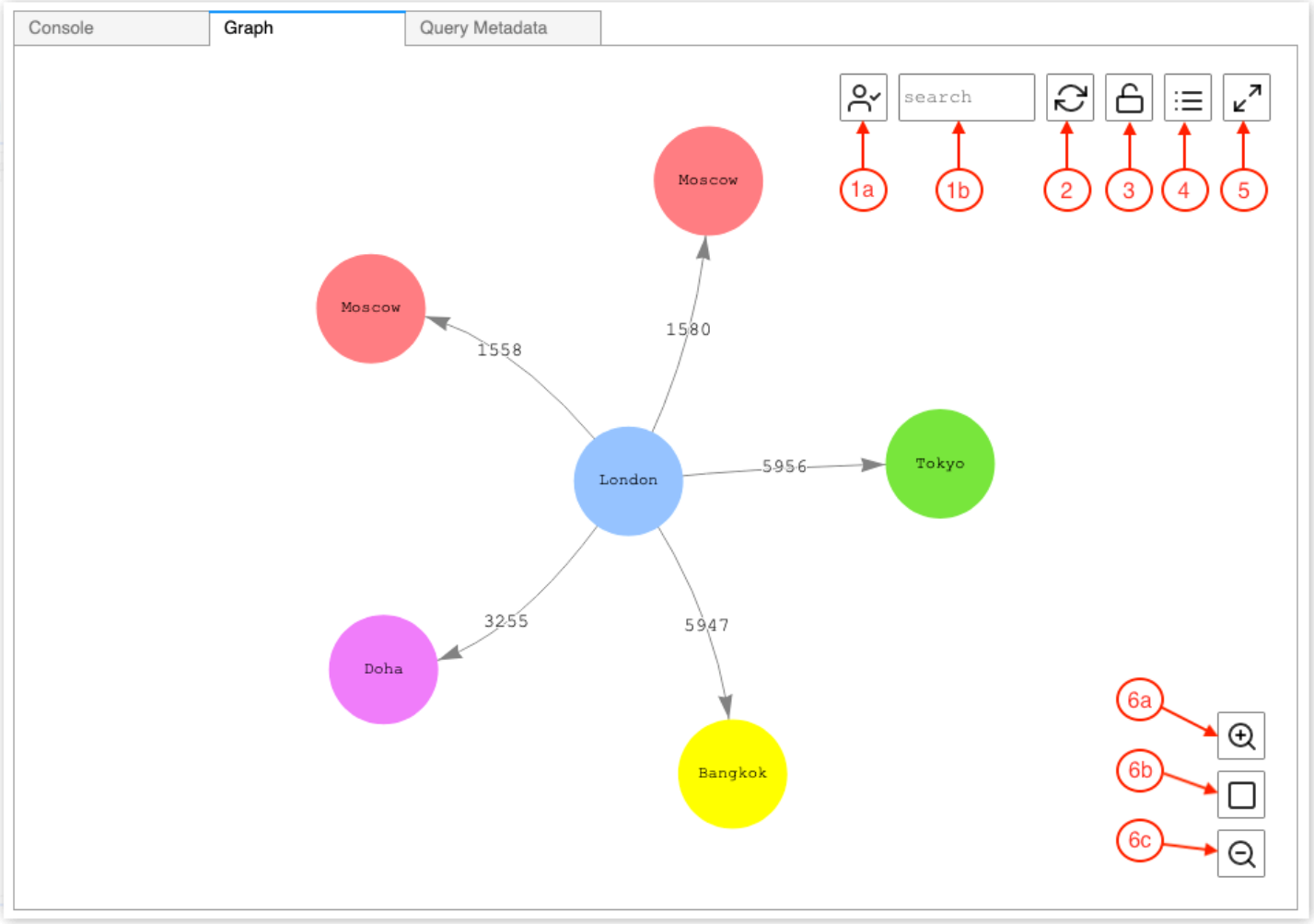
In molti casi Neptune Workbench può creare un diagramma visivo dei risultati della query e restituirli in formato tabulare. Nella scheda Grafo dei risultati della query è disponibile la visualizzazione grafo, se generata.

Oltre alle funzionalità di visualizzazione predefinite descritte qui, puoi anche utilizzare strumenti di visualizzazione più avanzati con i notebook per grafi Neptune.

> **Note**
>
> Per accedere alle funzionalità e alle correzioni aggiunte di recente nei notebook che già utilizzi, prima arresta e poi riavvia l'istanza notebook.

Panoramica dell'interfaccia della scheda Grafo

Questo diagramma identifica gli elementi dell'interfaccia utente presenti nella scheda Grafo:



1. Ricerca nel grafo
   a. Attiva/disattiva UUID: attiva o disattiva l'inclusione dei valori della proprietà ID nella ricerca nel grafo. Per impostazione predefinita, l'inclusione degli ID è abilitata. Se questa opzione è disattivata, le corrispondenze relative alle proprietà ID, incluse le proprietà degli archi che fanno riferimento agli ID dei nodi, non determinano l'evidenziazione degli elementi.
   b. Cerca nel campo di testo: evidenzia tutti i valori delle proprietà dei vertici e degli archi che contengono la stringa di testo specificata qui.
2. Reimpostazione del grafo: esegue nuovamente la simulazione fisica del grafo e imposta lo zoom per adattare il grafo alla finestra.
3. Attiva/disattiva fisica del grafo: attiva o disattiva l'esecuzione della simulazione fisica del grafo. La fisica è abilitata per impostazione predefinita e consente al grafo di cambiare in modo dinamico. Se è disabilitata, i vertici rimangono bloccati in posizione quando non vengono spostati altri vertici.

4. Visualizzazione dei dettagli: quando viene selezionato un nodo o un arco, viene visualizzato un elenco delle chiavi e dei valori delle proprietà dell'elemento, se disponibili nei risultati della query.
5. Visualizzazione schermo intero: espande la finestra della scheda Grafo per adattarla allo schermo. Facendo nuovamente clic la scheda Grafo viene ridotta.
6. Opzioni zoom
   a. Esegui lo zoom
   b. Reimpostazione zoom: imposta lo zoom per adattare tutti i vertici alla finestra della scheda Grafo.
   c. Zoom indietro

Visualizzazione dei risultati delle query Gremlin

Neptune Workbench crea una visualizzazione dei risultati della query per qualsiasi query Gremlin che restituisce un elemento `path`. Per visualizzarla, seleziona la scheda Grafo a destra della scheda Console sotto la query dopo averla eseguita.

È possibile utilizzare i suggerimenti per la visualizzazione delle query per controllare il modo in cui il visualizzatore genera i diagrammi dell'output della query. Questi suggerimenti seguono il comando magic di cella `%%gremlin` e sono preceduti dal nome del parametro `--path-pattern` (o dalla relativa forma breve, `-p`):

```
%%gremlin -p comma-separated hints
```

Puoi anche usare il flag `--group-by` (o `-g`) per specificare una proprietà dei vertici in base a cui raggrupparli. È possibile specificare un colore o un'icona per diversi gruppi di vertici.

I nomi dei suggerimenti riflettono i passaggi di Gremlin comunemente usati per l'attraversamento dei vertici e si comportano di conseguenza. È possibile utilizzare una combinazione di più suggerimenti, separati da virgole, senza spazi tra di essi. I suggerimenti utilizzati devono corrispondere ai passaggi di Gremlin corrispondenti nella query visualizzata. Ecco un esempio:

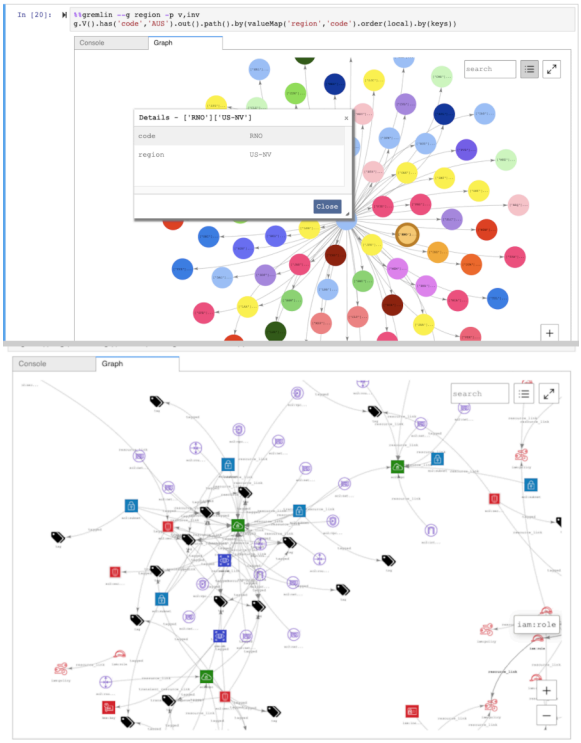
```
%%gremlin -p v,oute,inv
g.V().hasLabel('airport').outE().inV().path().by('code').by('dist').limit(5)
```

I suggerimenti di visualizzazione disponibili sono i seguenti:

```
v
```

```
inv
outv
e
ine
oute
```

Di seguito sono elencati alcuni esempi di visualizzazioni dei grafi che usano i gruppi:



Visualizzazione dei risultati delle query SPARQL

Neptune Workbench crea una visualizzazione dei risultati della query per tutte le query SPARQL che accettano uno di questi formati:

- `SELECT ?subject ?predicate ?object`
- `SELECT ?s ?p ?o`

Per visualizzarla, seleziona la scheda Grafo a destra della scheda Tabella sotto la query dopo averla eseguita.

Per impostazione predefinita, una visualizzazione SPARQL include solo modelli di triple in cui `o?` è un `uri` o un `bnode` (nodo vuoto). Tutti gli altri tipi di `?o` associazione, come stringhe letterali o numeri

interi, vengono trattati come proprietà del nodo `?s` che possono essere visualizzate nel riquadro Dettagli nella scheda Grafo.

In molti casi, tuttavia, potrebbe essere necessario includere tali valori letterali come vertici nella visualizzazione. A tale scopo, usa l'hint di query `--expand-all` dopo il comando magic di cella `%%sparql`:

```
%%sparql --expand-all
```

Questo indica al visualizzatore di includere tutti i risultati `?s ?p ?o` nel diagramma del grafo, indipendentemente dal tipo di associazione.

Puoi vedere questo suggerimento utilizzato in tutto il notebook `Air-Routes-SPARQL.ipynb` e puoi sperimentare eseguendo le query con e senza il suggerimento per osservare che differenza fa nella visualizzazione.

Accesso ai notebook con tutorial sulla visualizzazione in Neptune Workbench

I due notebook con tutorial sulla visualizzazione forniti in Neptune Workbench forniscono numerosi esempi in Gremlin e in SPARQL su come eseguire query efficaci sui dati dei grafi e visualizzare i risultati.

Passa ai notebook di visualizzazione

1. Nel riquadro di navigazione a sinistra scegli il pulsante Apri notebook a destra.
2. All'apertura di Neptune Workbench, con Jupyter in esecuzione, verrà visualizzata una cartella Neptune al livello superiore. Sceglierlo per aprire la cartella.
3. Al livello successivo è presente una cartella denominata 02-Visualization. Aprire la cartella. All'interno sono disponibili vari notebook che illustrano diversi modi per eseguire query dati dei grafi, in Gremlin e in SPARQL, e come visualizzare i risultati delle query:

   - Air-Routes-Gremlin
   - Air-Routes-SPARQL
   - Blog sulle visualizzazioni in Workbench
   - EPL-Gremlin
   - EPL-SPARQL

Seleziona un notebook per provare a usare le query al suo interno.

Configurazione di Neptune

Ti diamo il benvenuto in Amazon Neptune. Questa sezione aiuta a creare un nuovo cluster database Neptune e a trovare l'argomento che interessa nella documentazione di Neptune.

> **Note**
>
> Per le architetture di riferimento di database a AWS grafi e le architetture di distribuzione di riferimento, consulta Amazon Neptune Resources. Queste risorse possono aiutare a effettuare scelte riguardo ai modelli di dati a grafo e linguaggi di query, nonché accelerare il processo di sviluppo.

Argomenti

Scelta del tipo di istanza database Neptune giusto

Amazon Neptune offre diverse dimensioni e famiglie di istanze, che offrono funzionalità diverse adatte a diversi carichi di lavoro a grafo. In questa sezione è possibile scegliere il tipo di istanza migliore per le tue esigenze.

Per i prezzi di ogni tipo di istanza di queste famiglie, consulta la pagina dei prezzi di Neptune.

Panoramica dell'allocazione delle risorse delle istanze

Ogni tipo e dimensione di istanza Amazon EC2 utilizzata in Neptune offre una quantità definita di memoria di calcolo (vCPU) e di sistema. L'archiviazione primaria di Neptune è esterna alle istanze database in un cluster, il che consente di dimensionare la capacità di calcolo e di archiviazione indipendentemente l'una dall'altra.

Questa sezione si concentra su come dimensionare le risorse di calcolo e sulle differenze tra ciascuna delle varie famiglie di istanze.

In tutte le famiglie di istanze, le risorse vCPU sono allocate per supportare due (2) thread di esecuzione di query per vCPU. Questo supporto è dettato dalla dimensione dell'istanza. Per determinare la dimensione corretta di una determinata istanza database di Neptune, è necessario considerare la possibile concorrenza dell'applicazione e la latenza media delle query. È possibile stimare il numero di vCPU necessarie nel modo seguente, dove la latenza viene misurata come latenza media delle query in secondi e la concorrenza viene misurata come numero target di query al secondo:

$$vCPUs = \frac{latency \times concurrency}{2}$$

> **Note**
>
> Le query SPARQL, le query openCypher e le query di lettura Gremlin che utilizzano il motore di query DFE possono, in determinate circostanze, utilizzare più di un thread di esecuzione per query. Quando si dimensiona inizialmente il cluster database, si parte dal presupposto che ogni query consumerà un singolo thread per esecuzione che può essere aumentato se si osserva una contropressione nella coda di query. Questo può essere osservato utilizzando le `/sparql/status` API `/gremlin/status/oc/status`, o oppure può essere osservato anche utilizzando la metrica. `MainRequestsPendingRequestsQueue` CloudWatch

La memoria di sistema di ogni istanza è suddivisa in due allocazioni principali: cache del pool di buffer e memoria dei thread di esecuzione delle query.

Circa due terzi della memoria disponibile in un'istanza sono allocati per la cache del pool di buffer. La cache del pool di buffer viene utilizzata per memorizzare nella cache i componenti del grafo utilizzati

più di recente per un accesso più rapido alle query che accedono ripetutamente a tali componenti. Le istanze con una maggiore quantità di memoria di sistema dispongono di cache del pool di buffer più grandi in grado di archiviare localmente una parte maggiore del grafo. Un utente può effettuare l'ottimizzazione della quantità appropriata di cache del buffer-pool monitorando le metriche relative agli accessi e gli errori della cache buffer disponibili in. CloudWatch

Può essere necessario aumentare la dimensione dell'istanza se la percentuale di riscontri nella cache scende al di sotto del 99,9% per un periodo di tempo costante. Ciò suggerisce che il pool di buffer non è sufficientemente grande e che il motore deve recuperare i dati dal volume di archiviazione sottostante più spesso di quanto sia efficiente.

Il terzo restante della memoria di sistema è distribuito in modo uniforme tra i thread di esecuzione delle query, con una parte di memoria residua per il sistema operativo e un piccolo pool dinamico che i thread possono utilizzare in base alle esigenze. La memoria disponibile per ogni thread aumenta leggermente da una dimensione di istanza all'altra fino a un tipo di istanza `8xl`, dimensione alla quale la memoria allocata per thread raggiunge il massimo.

Il momento di aggiungere altra memoria ai thread è quando si verifica un'eccezione `OutOfMemoryException` (OOM). Le eccezioni OOM si verificano quando un thread richiede una quantità di memoria superiore a quella massima ad esso allocata. Ciò non significa che l'intera istanza stia esaurendo la memoria.

Tipi di istanza **`t3`** e **`t4g`**

La famiglia di istanze `t3` e `t4g` offre un'opzione a basso costo per iniziare a utilizzare un database a grafo e anche per lo sviluppo e i test iniziali. Queste istanze sono idonee per l'offerta gratuita di Neptune, che consente ai nuovi clienti di utilizzare Neptune gratuitamente per le prime 750 ore di istanza utilizzate in un account AWS autonomo o raggruppate presso un'organizzazione con fatturazione consolidata (account pagante). AWS

Le istanze `t3` e `t4g` sono disponibili solo nella configurazione di medie dimensioni (`t3.medium` e `t4g.medium`).

Non sono destinate all'uso in un ambiente di produzione.

Poiché queste istanze dispongono di risorse molto limitate, non sono consigliate per testare il tempo di esecuzione delle query o le prestazioni complessive del database. Per valutare le prestazioni delle query, esegui l'aggiornamento a una delle altre famiglie di istanze.

Famiglia di tipi di istanza **r4**

OBSOLETA: la famiglia `r4` è stata offerta al momento del lancio di Neptune nel 2018, ma ora i tipi di istanza più recenti offrono un rapporto prezzo/prestazioni molto migliore. A partire dalla versione 1.1.0.0 del motore, Neptune non supporta più i tipi di istanza `r4`.

Famiglia di tipi di istanza **r5**

La famiglia `r5` contiene tipi di istanza ottimizzata per la memoria che funzionano bene per la maggior parte dei casi d'uso a grafo. La famiglia `r5` contiene tipi di istanza da `r5.large` a `r5.24xlarge`. Le prestazioni di calcolo di questi tipi di istanza aumentano in modo lineare con l'aumentare delle dimensioni. Ad esempio, un'istanza `r5.xlarge` (4 vCPU e 32 GiB di memoria) ha il doppio delle vCPU e della memoria di un'istanza `r5.large` (2 vCPU e 16 GiB di memoria) e un'istanza `r5.2xlarge` (8 vCPU e 64 GiB di memoria) ha il doppio delle vCPU e della memoria di un'istanza `r5.xlarge`. Le prestazioni delle query aumentano direttamente con la capacità di calcolo fino al tipo di istanza `r5.12xlarge`.

La famiglia di istanze `r5` ha un'architettura CPU Intel a 2 socket. Il tipo `r5.12xlarge` e quelli più piccoli utilizzano un socket singolo e la memoria di sistema di proprietà di quel processore a socket singolo. I tipi `r5.16xlarge` e `r5.24xlarge` utilizzano entrambi i socket e la memoria disponibile. Poiché è necessario un certo sovraccarico di gestione della memoria tra due processori fisici in un'architettura a 2 socket, i vantaggi in termini di prestazioni che si ottengono passando da un tipo di istanza `r5.12xlarge` a un tipo di istanza `r5.16xlarge` o `r5.24xlarge` non sono così lineari come quelli che si ottengono passando a istanze di dimensioni inferiori.

Famiglia di tipi di istanza **r5d**

Neptune dispone di una funzionalità di cache di ricerca che può essere utilizzata per migliorare le prestazioni delle query che devono recuperare e restituire un gran numero di valori di proprietà e valori letterali. Questa funzionalità viene utilizzata principalmente dai clienti con query che devono restituire molti attributi. La cache di ricerca migliora le prestazioni di queste query recuperando questi valori degli attributi localmente anziché cercarli ripetutamente nell'archiviazione indicizzata di Neptune.

La cache di ricerca viene implementata utilizzando un volume EBS collegato a NVMe su un tipo di istanza `r5d`. Viene abilitata utilizzando il gruppo di parametri di un cluster. Man mano che i dati vengono recuperati dall'archiviazione indicizzata di Neptune, i valori delle proprietà e i valori letterali RDF vengono memorizzati nella cache all'interno di questo volume NVMe.

Se non è necessaria la funzionalità di cache di ricerca, utilizza un tipo di istanza `r5` standard anziché un tipo `r5d`, per evitare il costo più elevato del tipo `r5d`

La famiglia `r5d` ha tipi di istanza delle stesse dimensioni della famiglia `r5`, da `r5d.large` a `r5d.24xlarge`.

Famiglia di tipi di istanza **`r6g`**

AWS ha sviluppato un proprio processore basato su ARM chiamato Graviton, che offre un rapporto prezzo/prestazioni migliore rispetto agli equivalenti Intel e AMD. La famiglia `r6g` utilizza il processore Graviton2. Nei nostri test, il processore Graviton2 offre prestazioni migliori del 10-20% per le query a grafo di tipo OLTP (vincolate). Le query più grandi, di tipo OLAP, tuttavia, potrebbero essere leggermente meno performanti con i processori Graviton2 rispetto a quelli Intel a causa delle prestazioni di paging della memoria leggermente inferiori.

È anche importante notare che la famiglia `r6g` ha un'architettura a socket singolo, il che significa che le prestazioni aumentano in modo lineare con la capacità di calcolo da un tipo `r6g.large` a un tipo `r6g.16xlarge` (il tipo più grande della famiglia).

Famiglia di tipi di istanza **`r6i`**

Le istanze Amazon R6i sono basate su processori scalabili Intel Xeon di terza generazione (nome in codice Ice Lake) e sono la soluzione ideale per carichi di lavoro che richiedono molta memoria. In generale, offrono prestazioni fino al 15% migliori in termini di prezzo di calcolo e fino al 20% in più di larghezza di banda di memoria per vCPU rispetto ai tipi di istanza R5 comparabili.

Famiglia di tipi di istanza **`x2g`**

Alcuni casi d'uso a grafo ottengono prestazioni migliori quando le istanze dispongono di cache del pool di buffer più grandi. La famiglia `x2g` è stata lanciata per supportare meglio questi casi d'uso. La `x2g` famiglia ha un rapporto memory-to-v CPU maggiore rispetto alla famiglia or. `r5` `r6g` Anche le istanze `x2g` utilizzano il processore Graviton2 e hanno molte delle stesse caratteristiche prestazionali dei tipi di istanza `r6g`, oltre a una cache del pool di buffer più grande.

Se si usano tipi di istanza `r5` o `r6g` con un basso utilizzo della CPU e un'elevata percentuale di mancati riscontri nella cache del pool di buffer, provare invece a utilizzare la famiglia `x2g`. In questo modo, otterrai la memoria aggiuntiva di cui hai bisogno senza dover pagare per una maggiore capacità della CPU.

Tipo di istanza `serverless`

La funzionalità Neptune Serverless può dimensionare le dimensioni delle istanze in modo dinamico in base alle esigenze di risorse del carico di lavoro. Invece di calcolare quante vCPU sono necessarie per l'applicazione, Neptune Serverless consente di impostare limiti inferiori e superiori per la capacità di calcolo (misurata in unità di capacità Neptune) per le istanze del cluster database. I carichi di lavoro con utilizzo variabile possono essere ottimizzati in termini di costi utilizzando istanze serverless anziché istanze con provisioning.

È possibile configurare istanze con provisioning e serverless nello stesso cluster database per ottenere una configurazione ottimale in termini di costi e prestazioni.

Scelta del tipo di archiviazione corretto per il cluster di database Neptune

Neptune offre due tipi di archiviazione con un modello di prezzi differente:

- Archiviazione standard: l'archiviazione standard fornisce archiviazione di database conveniente per applicazioni con un numero di operazioni di I/O da moderato a basso.
- Storage ottimizzato per l'I/O — Con lo storage ottimizzato per l'I/O, disponibile dalla versione 1.3.0.0 del motore, paghi solo per lo storage e le istanze che utilizzi. I costi di archiviazione sono più elevati rispetto all'archiviazione standard e non è previsto alcun pagamento per l'I/O utilizzato. Se il numero di operazioni di I/O è elevato, l'archiviazione capacità di IOPS allocata può ridurre i costi in modo significativo.

  L'archiviazione ottimizzata per l'I/O è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O a un costo prevedibile, con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente. Puoi passare dal tipo di storage ottimizzato per I/O a quello standard solo una volta ogni 30 giorni.

  Per informazioni sui prezzi dell'archiviazione ottimizzata per l'I/O, consulta la pagina Prezzi di Amazon Neptune. Nella sezione seguente viene descritto come configurare l'archiviazione ottimizzata per l'I/O per un cluster di database Neptune.

Scelta dell'archiviazione ottimizzata per l'I/O per un cluster di database Neptune

Per impostazione predefinita, i cluster di database Neptune utilizzano l'archiviazione standard. Puoi abilitare l'archiviazione ottimizzata per l'I/O su un cluster di database al momento della creazione, come descritto di seguito:

Di seguito è riportato un esempio di come abilitare l'archiviazione ottimizzata per l'I/O quando si crea un cluster utilizzando la AWS CLI:

```
aws neptune create-db-cluster \
  --database-name (name for the new database) \
  --db-cluster-identifier (an ID for the cluster) \
  --engine neptune \
  --engine-version 1.3.0.0 \
  --storage-type iopt1
```

Quindi, l'archiviazione ottimizzata per l'I/O è abilitata per qualsiasi istanza creata automaticamente:

```
aws neptune create-db-instance \
  --db-cluster-identifier (the ID of the new cluster) \
  --db-instance-identifier (an ID for the new instance) \
  --engine neptune \
  --db-instance-class db.r5.large
```

Puoi anche modificare un cluster di database esistente per abilitare l'archiviazione ottimizzata per l'I/O, in questo modo:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (the ID of a cluster without I/O-Optimized storage) \
  --storage-type iopt1 \
  --apply-immediately
```

Puoi ripristinare uno snapshot di backup su un cluster di database con l'archiviazione ottimizzata per l'I/O abilitata:

```
aws neptune restore-db-cluster-from-snapshot \
  --db-cluster-identifier (an ID for the restored cluster) \
  --snapshot-identifier (the ID of the snapshot to restore from) \
  --engine neptune \
```

```
  --engine-version 1.3.0.0 \
  --storage-type iopt1
```

Puoi determinare se un cluster utilizza l'archiviazione ottimizzata per l'I/O utilizzando qualsiasi chiamata `describe-`. Se l'archiviazione ottimizzata per l'I/O è abilitata, la chiamata restituisce un campo di tipo di archiviazione impostato su `iop1`.

Creazione di un nuovo cluster database Neptune

Il modo più semplice per creare un nuovo cluster Amazon Neptune DB consiste nell'utilizzare AWS CloudFormation un modello che crei tutte le risorse necessarie per te, senza dover fare tutto a mano. Il AWS CloudFormation modello esegue gran parte della configurazione per te, inclusa la creazione di un'istanza Amazon Elastic Compute Cloud (Amazon EC2):

Per avviare un nuovo cluster Neptune DB utilizzando un modello AWS CloudFormation

1. Creare un nuovo utente IAM con le autorizzazioni necessarie per lavorare con il cluster database Neptune, come spiegato in Autorizzazioni degli utenti IAM.
2. Imposta i prerequisiti aggiuntivi necessari per utilizzare il AWS CloudFormation modello, come spiegato in. Prerequisiti per l'utilizzo AWS CloudFormation per configurare Neptune
3. Richiama lo AWS CloudFormation stack, come descritto in. Utilizzo di uno AWS CloudFormation stack per creare un cluster Neptune DB

Puoi anche creare un database globale di Neptune che si estende su Regioni AWS più pagine, abilitando letture globali a bassa latenza e fornendo un ripristino rapido nel raro caso in cui un'interruzione riguardi un intero sistema. Regione AWS

Per informazioni sulla creazione manuale di un cluster Amazon Neptune utilizzando AWS Management Console il, consulta. Avvio di un cluster database Neptune mediante la console AWS Management Console

Puoi anche usare un AWS CloudFormation modello per creare una funzione Lambda da usare con Neptune (vedi). Utilizzo di AWS CloudFormation per creare una funzione Lambda da usare in Neptune

Per informazioni generali sulla gestione di cluster e istanze in Neptune, consulta Gestione del database Amazon Neptune.

Prerequisiti per l'utilizzo AWS CloudFormation per configurare Neptune

Prima di creare un cluster Amazon Neptune utilizzando AWS CloudFormation un modello, è necessario disporre di quanto segue:

- Coppia di chiavi Amazon EC2.
- Le autorizzazioni necessarie per l'utilizzo. AWS CloudFormation

Crea una coppia di chiavi Amazon EC2 da utilizzare per il lancio di un cluster Neptune utilizzando AWS CloudFormation

Per avviare un cluster Neptune DB utilizzando un modello, devi avere AWS CloudFormation una coppia di chiavi Amazon EC2 (e il relativo file PEM associato) disponibile nella regione in cui crei lo stack. AWS CloudFormation

Per creare la coppia di chiavi, consulta la sezione Creazione di una coppia di chiavi tramite Amazon EC2 nella Guida per l'utente di Amazon EC2 per le istanze Linux o Creazione di una coppia di chiavi tramite Amazon EC2 nella Guida per l'utente di Amazon EC2 per le istanze Windows per le istruzioni.

Aggiungi le politiche IAM per concedere le autorizzazioni necessarie per utilizzare il modello AWS CloudFormation

Innanzitutto, è necessario avere un utente IAM configurato con le autorizzazioni necessarie per lavorare con Neptune, come descritto in Creazione di un utente IAM con autorizzazioni per Neptune.

Quindi devi aggiungere la policy AWS gestita a quell'utente.
`AWSCloudFormationReadOnlyAccess`

Infine, è necessario creare la seguente policy gestita dal cliente e aggiungerla a tale utente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "rds:CreateDBCluster",
          "rds:CreateDBInstance"
      ],
      "Resource": [
```

```
            "arn:aws:rds:*:*:*"
        ],
        "Condition": {
            "StringEquals": {
                "rds:DatabaseEngine": ["graphdb","neptune"]
            }
        }
    },
    {
        "Action": [
            "rds:AddRoleToDBCluster",
            "rds:AddSourceIdentifierToSubscription",
            "rds:AddTagsToResource",
            "rds:ApplyPendingMaintenanceAction",
            "rds:CopyDBClusterParameterGroup",
            "rds:CopyDBClusterSnapshot",
            "rds:CopyDBParameterGroup",
            "rds:CreateDBClusterParameterGroup",
            "rds:CreateDBClusterSnapshot",
            "rds:CreateDBParameterGroup",
            "rds:CreateDBSubnetGroup",
            "rds:CreateEventSubscription",
            "rds:DeleteDBCluster",
            "rds:DeleteDBClusterParameterGroup",
            "rds:DeleteDBClusterSnapshot",
            "rds:DeleteDBInstance",
            "rds:DeleteDBParameterGroup",
            "rds:DeleteDBSubnetGroup",
            "rds:DeleteEventSubscription",
            "rds:DescribeAccountAttributes",
            "rds:DescribeCertificates",
            "rds:DescribeDBClusterParameterGroups",
            "rds:DescribeDBClusterParameters",
            "rds:DescribeDBClusterSnapshotAttributes",
            "rds:DescribeDBClusterSnapshots",
            "rds:DescribeDBClusters",
            "rds:DescribeDBEngineVersions",
            "rds:DescribeDBInstances",
            "rds:DescribeDBLogFiles",
            "rds:DescribeDBParameterGroups",
            "rds:DescribeDBParameters",
            "rds:DescribeDBSecurityGroups",
            "rds:DescribeDBSubnetGroups",
            "rds:DescribeEngineDefaultClusterParameters",
```

```
        "rds:DescribeEngineDefaultParameters",
        "rds:DescribeEventCategories",
        "rds:DescribeEventSubscriptions",
        "rds:DescribeEvents",
        "rds:DescribeOptionGroups",
        "rds:DescribeOrderableDBInstanceOptions",
        "rds:DescribePendingMaintenanceActions",
        "rds:DescribeValidDBInstanceModifications",
        "rds:DownloadDBLogFilePortion",
        "rds:FailoverDBCluster",
        "rds:ListTagsForResource",
        "rds:ModifyDBCluster",
        "rds:ModifyDBClusterParameterGroup",
        "rds:ModifyDBClusterSnapshotAttribute",
        "rds:ModifyDBInstance",
        "rds:ModifyDBParameterGroup",
        "rds:ModifyDBSubnetGroup",
        "rds:ModifyEventSubscription",
        "rds:PromoteReadReplicaDBCluster",
        "rds:RebootDBInstance",
        "rds:RemoveRoleFromDBCluster",
        "rds:RemoveSourceIdentifierFromSubscription",
        "rds:RemoveTagsFromResource",
        "rds:ResetDBClusterParameterGroup",
        "rds:ResetDBParameterGroup",
        "rds:RestoreDBClusterFromSnapshot",
        "rds:RestoreDBClusterToPointInTime"
      ],
      "Effect": "Allow",
      "Resource": [
          "*"
      ]
    },
    {
      "Action": [
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:ListMetrics",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcs",
        "kms:ListAliases",
```

```
        "kms:ListKeyPolicies",
        "kms:ListKeys",
        "kms:ListRetirableGrants",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents",
        "sns:ListSubscriptions",
        "sns:ListTopics",
        "sns:Publish"
      ],
      "Effect": "Allow",
      "Resource": [
          "*"
      ]
    },
    {
      "Action": "iam:PassRole",
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
            "iam:passedToService": "rds.amazonaws.com"
        }
      }
    },
    {
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
      "Condition": {
        "StringLike": {
            "iam:AWSServiceName": "rds.amazonaws.com"
        }
      }
    }
  ]
}
```

> **Note**
>
> Le seguenti autorizzazioni sono richieste solo per eliminare uno stack: `iam:DeleteRole`, `iam:RemoveRoleFromInstanceProfile`, `iam:DeleteRolePolicy`, `iam:DeleteInstanceProfile` e `ec2:DeleteVpcEndpoints`.

Inoltre, `ec2:*Vpc` concede le autorizzazioni `ec2:DeleteVpc`.

Utilizzo di uno AWS CloudFormation stack per creare un cluster Neptune DB

È possibile utilizzare un AWS CloudFormation modello per configurare un cluster Neptune DB.

1. Per avviare lo AWS CloudFormation stack sulla AWS CloudFormation console, scegli uno dei pulsanti Launch Stack nella tabella seguente.

Regione	Vista	Visualizzazione in Designer	Avvia
Stati Uniti orientali (Virginia settentrionale)	Visualizzazione	Visualizzazione in Designer	
Stati Uniti orientali (Ohio)	Visualizzazione	Visualizzazione in Designer	
Stati Uniti occidentali (California settentrionale)	Visualizzazione	Visualizzazione in Designer	
US West (Oregon)	Visualizzazione	Visualizzazione in Designer	
Canada (Centrale)	Visualizzazione	Visualizzazione in Designer	
Sud America (San Paolo)	Visualizzazione	Visualizzazione in Designer	
Europa (Stoccolma)	Visualizzazione	Visualizzazione in Designer	
Europa (Irlanda)	Visualizzazione	Visualizzazione in Designer	
Europa (Londra)	Visualizzazione	Visualizzazione in Designer	

Regione	Vista	Visualizzazione in Designer	Avvia
Europa (Parigi)	Visualizzazione	Visualizzazione in Designer	
Europa (Francoforte)	Visualizzazione	Visualizzazione in Designer	
Medio Oriente (Bahrein)	Visualizzazione	Visualizzazione in Designer	
Medio Oriente (Emirati Arabi Uniti)	Visualizzazione	Visualizzazione in Designer	
Israele (Tel Aviv)	Visualizzazione	Visualizzazione in Designer	
Africa (Città del Capo)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Hong Kong)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Tokyo)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Seul)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Singapore)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Sydney)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Mumbai)	Visualizzazione	Visualizzazione in Designer	

Regione	Vista	Visualizzazione in Designer	Avvia
Cina (Pechino)	Visualizzazione	Visualizzazione in Designer	
Cina (Ningxia)	Visualizzazione	Visualizzazione in Designer	
AWS GovCloud (Stati Uniti occidentali)	Visualizzazione	Visualizzazione in Designer	
AWS GovCloud (Stati Uniti orientali)	Visualizzazione	Visualizzazione in Designer	

2. Nella pagina Select Template (Seleziona modello), selezionare Next (Avanti).
3. Nella pagina Specify Details, scegli una key pair per KeyPairNameEC2SSH.

   Questa coppia di chiavi è obbligatoria per accedere all'istanza EC2. Assicurati di disporre del file PEM per la coppia di chiavi da te scelta.
4. Seleziona Avanti.
5. Nella pagina Opzioni, scegli Avanti.
6. Nella pagina Revisione, seleziona la prima casella di controllo per accettare la creazione delle risorse IAM da parte di AWS CloudFormation . Seleziona la seconda casella di controllo per confermare `CAPABILITY_AUTO_EXPAND` per il nuovo stack.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` conferma in modo esplicito che, durante la creazione dello stack, le macro verranno ampliate senza revisione preventiva. Gli utenti spesso creano un set di modifiche da un modello elaborato, quindi le modifiche apportate dalle macro possono essere riesaminate prima dell'effettiva creazione dello stack. Per ulteriori informazioni, consulta l'API AWS CloudFormation CreateStack.

   Quindi, scegli Crea.

> **Note**
>
> È inoltre possibile utilizzare il AWS CloudFormation modello per aggiornare la versione del motore del cluster DB.

Configurare l'Amazon VPC in cui si trova il cluster database Amazon Neptune

Un cluster di database Amazon Neptune può essere creato solo in un Amazon Virtual Private Cloud (Amazon VPC). I relativi endpoint sono accessibili all'interno di tale VPC.

Esistono diversi modi per configurare il VPC, a seconda di come desideri accedere al cluster database.

Di seguito sono elencati alcuni aspetti da considerare quando si configura il VPC in cui si trova il cluster database Neptune:

- Il VPC deve avere almeno due sottoreti. Queste sottoreti devono trovarsi in due diverse zone di disponibilità (AZ). Distribuendo le istanze cluster in almeno due zone di disponibilità, Neptune garantisce che siano sempre disponibili istanze nel cluster database anche nell'improbabile caso di errore di una zona di disponibilità. Il volume del cluster database Neptune si estende sempre su tre zone di disponibilità per fornire un'archiviazione durevole con una probabilità estremamente bassa di perdita di dati.
- I blocchi CIDR di ogni sottorete devono essere sufficientemente grandi da fornire gli indirizzi IP necessari a Neptune durante le attività di manutenzione, il failover e il dimensionamento.
- Il VPC deve avere un gruppo di sottoreti del database contenente le sottoreti create. Neptune sceglie una delle sottoreti del gruppo di sottoreti e un indirizzo IP all'interno di tale sottorete da associare a ciascuna istanza database del cluster database. L'istanza database si trova quindi nella stessa zona di disponibilità della sottorete.
- Il VPC deve avere il DNS abilitato (sia i nomi host DNS che la risoluzione DNS).
- Il VPC deve avere un gruppo di sicurezza VPC per consentire l'accesso al cluster database.
- La tenancy in un VPC Neptune deve essere impostata su Default.

Aggiunta di sottoreti al VPC in cui si trova il cluster database Neptune

una sottorete è un intervallo di indirizzi IP nel VPC; È possibile avviare le risorse, ad esempio un cluster database Neptune o un'istanza EC2, in una sottorete specifica. Quando si crea una sottorete, è necessario specificare il blocco CIDR IPv4 per la sottorete, che è un sottoinsieme del blocco CIDR del VPC. Ogni sottorete deve risiedere totalmente all'interno di una zona di disponibilità (AZ) e non può estendersi in altre zone. Avviando le istanze in zone di disponibilità separate, sarà possibile proteggere le applicazioni da un errore in una delle zone. Per ulteriori informazioni, consulta la documentazione delle sottoreti VPC.

Un cluster database Neptune richiede almeno due sottoreti VPC.

Per aggiungere sottoreti a un VPC

1. Accedi AWS Management Console e apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nel pannello di navigazione, scegli Subnets (Sottoreti).
3. Nel Pannello di controllo VPC, scegliere Sottoreti e quindi scegliere Crea sottorete.
4. Nella pagina Crea sottorete, scegliere il VPC in cui creare la sottorete.
5. In Impostazioni sottorete, effettuare le seguenti scelte:
    a. Inserire un nome per la nuova sottorete in Nome sottorete.
    b. Scegliere una zona di disponibilità (AZ) per la sottorete o lasciare la scelta su Nessuna preferenza.
    c. Inserire il blocco di indirizzi IP della sottorete in Blocco CIDR IPv4.
    d. Se necessario, aggiungere tag alla sottorete.
    e. Scegliere
6. Se si desidera creare contemporaneamente un'altra sottorete, scegliere Aggiungi nuova sottorete.
7. Scegliere Crea sottorete per creare le nuove sottoreti.

Creare un grippo di sottorete nel VPC

Creare un gruppo di sottoreti.

Per creare un gruppo di sottoreti Neptune

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Scegliere Subnet groups (Gruppi di sottoreti), quindi fare clic su Create DB Subnet Group (Crea gruppo di sottoreti database).
3. Immettere un nome e una descrizione per il nuovo gruppo di sottoreti (la descrizione è obbligatoria).
4. In VPC, scegliere il VPC in cui posizionare questo gruppo di sottoreti.
5. In Zona di disponibilità, scegliere la zona di disponibilità in cui posizionare questo gruppo di sottoreti.
6. In Sottorete, aggiungere una o più sottoreti di questa zona di disponibilità a questo gruppo di sottoreti.
7. Fare clic su Crea per creare il nuovo gruppo di sottoreti.

Creare un gruppo di sicurezza tramite la console VPC

I gruppi di sicurezza forniscono l'accesso al cluster database Neptune nel VPC. Fungono da firewall per il cluster database associato, controllando il traffico sia in entrata che in uscita a livello di istanza. Per impostazione predefinita, un'istanza database viene creata con un firewall e un gruppo di sicurezza predefinito che ne impedisce l'accesso. Per abilitare l'accesso, è necessario avere un gruppo di sicurezza VPC con regole aggiuntive.

La procedura seguente illustra come aggiungere una regola TCP personalizzata che specifichi l'intervallo di porte e gli indirizzi IP utilizzati dall'istanza Amazon EC2 per accedere al cluster database Neptune. È possibile utilizzare il gruppo di sicurezza VPC assegnato all'istanza EC2 anziché il relativo indirizzo IP.

Per creare un gruppo di sicurezza VPC per Neptune sulla console

1. Accedi AWS Management Console e apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nell'angolo in alto a destra della console, scegli la AWS regione in cui desideri creare un gruppo di sicurezza VPC per Neptune. Nell'elenco delle risorse Amazon VPC per la regione specifica verrà indicato che si dispone di almeno un VPC e di diverse sottoreti. Se non è indicato, non si dispone di un VPC predefinito in quella regione.

3. Nel riquadro di navigazione, in Sicurezza, scegliere Gruppi di sicurezza.
4. Scegliere Create Security Group (Crea gruppo di sicurezza). Nella finestra Crea gruppo di sicurezza, inserire il Nome del gruppo di sicurezza, una Descrizione e l'identificatore del VPC in cui risiederà il cluster database Neptune.
5. Aggiungere una regola in entrata per il gruppo di sicurezza di un'istanza Amazon EC2 da connettere al cluster database Neptune:
   a. Nell'area Regole in entrata, scegliere Aggiungi regola.
   b. Nell'elenco Tipo, lasciare selezionato Regola TCP personalizzata.
   c. Nella casella di testo Intervallo porte inserire 8182, il valore predefinito della porta per Neptune.
   d. In Origine, inserire l'intervallo di indirizzi IP (valore CIDR) da cui si accederà a Neptune o scegliere il nome di un gruppo di sicurezza esistente.
   e. Se occorre aggiungere altri indirizzi IP o intervalli di porta diversi, scegliere nuovamente Aggiungi regola.
6. Nell'area Regole in uscita, è possibile anche aggiungere una o più regole in uscita, se necessario.
7. Al termine, scegliere Create security group (Crea gruppo di sicurezza).

È possibile utilizzare questo nuovo gruppo di sicurezza VPC quando si crea un nuovo cluster database Neptune.

Se utilizzi un VPC predefinito, viene creato un gruppo di sottoreti predefinito per te che include tutte le sottoreti VPC. Quando si sceglie Crea database nella console Neptune, viene utilizzato il VPC predefinito a meno che non se ne specifichi uno diverso.

Assicurarsi di disporre del supporto DNS nel VPC

Domain Name System (DNS) è uno standard che consente di risolvere i nomi utilizzati su Internet nei corrispondenti indirizzi IP. Un nome host DNS assegna un nome a un computer in modo univoco ed è costituito da un nome host e un nome di dominio. I server DNS risolvono i nomi host DNS nei corrispondenti indirizzi IP.

Verificare che i nomi host DNS e la risoluzione DNS siano entrambi abilitati nel VPC. Gli attributi di rete VPC `enableDnsHostnames` e `enableDnsSupport` devono essere impostati su

`true`. Per visualizzare e modificare questi attributi, accedi alla console VPC all'indirizzo https://console.aws.amazon.com/vpc/.

Per ulteriori informazioni, consulta Utilizzo del DNS con il tuo VPC.

> **Note**
>
> Se si utilizza Route 53, verificare che la configurazione non sostituisca gli attributi di rete DNS nel VPC.

Connessione al grafo Amazon Neptune

Dopo aver creato un cluster database Neptune, il passaggio successivo consiste nell'impostare le modalità di connessione a esso.

Impostazione di **`curl`** o awscurl per la comunicazione con l'endpoint Neptune

Avere uno strumento a riga di comando per inviare query al cluster database Neptune è molto utile, come illustrato in molti esempi di questa documentazione. Lo strumento a riga di comando curl è un'opzione eccellente per comunicare con gli endpoint Neptune quando l'autenticazione IAM non è abilitata. Le versioni a partire dalla 7.75.0 supportano l'opzione `--aws-sigv4` per la firma delle richieste quando l'autenticazione IAM è abilitata.

Per gli endpoint in cui è abilitata l'autenticazione IAM, è possibile utilizzare anche awscurl, che utilizza quasi esattamente la stessa sintassi di `curl` ma supporta la firma delle richieste come richiesto dall'autenticazione IAM. A causa della maggiore sicurezza fornita dall'autenticazione IAM, in genere è una buona idea abilitarla.

Per informazioni su come usare `curl` (o `awscurl`), consulta la pagina man di curl e scarica il libro Everything curl.

Per connettersi tramite HTTPS (richiesto da Neptune), `curl` richiede l'accesso ai certificati appropriati. Se `curl` è in grado di individuare i certificati adeguati, gestisce le connessioni HTTPS in modo analogo alle connessioni HTTP, senza parametri aggiuntivi. Lo stesso vale per `awscurl`. Gli esempi di questa documentazione si basano su questo scenario.

Per scoprire come ottenere tali certificati e come formattarli correttamente in un archivio di certificati CA (Certificate Authority) che `curl` può utilizzare, consulta Verifica certificati SSL nella documentazione di `curl`.

È possibile specificare il percorso di questo archivio di certificati CA utilizzando la variabile di ambiente `CURL_CA_BUNDLE`. In Windows, `curl` cerca automaticamente un file denominato `curl-ca-bundle.crt`. Cerca prima nella stessa cartella di `curl.exe`, quindi in altri punti del percorso. Per ulteriori informazioni, consulta Verifica certificati SSL.

Diversi modi per connettersi a un cluster database Neptune

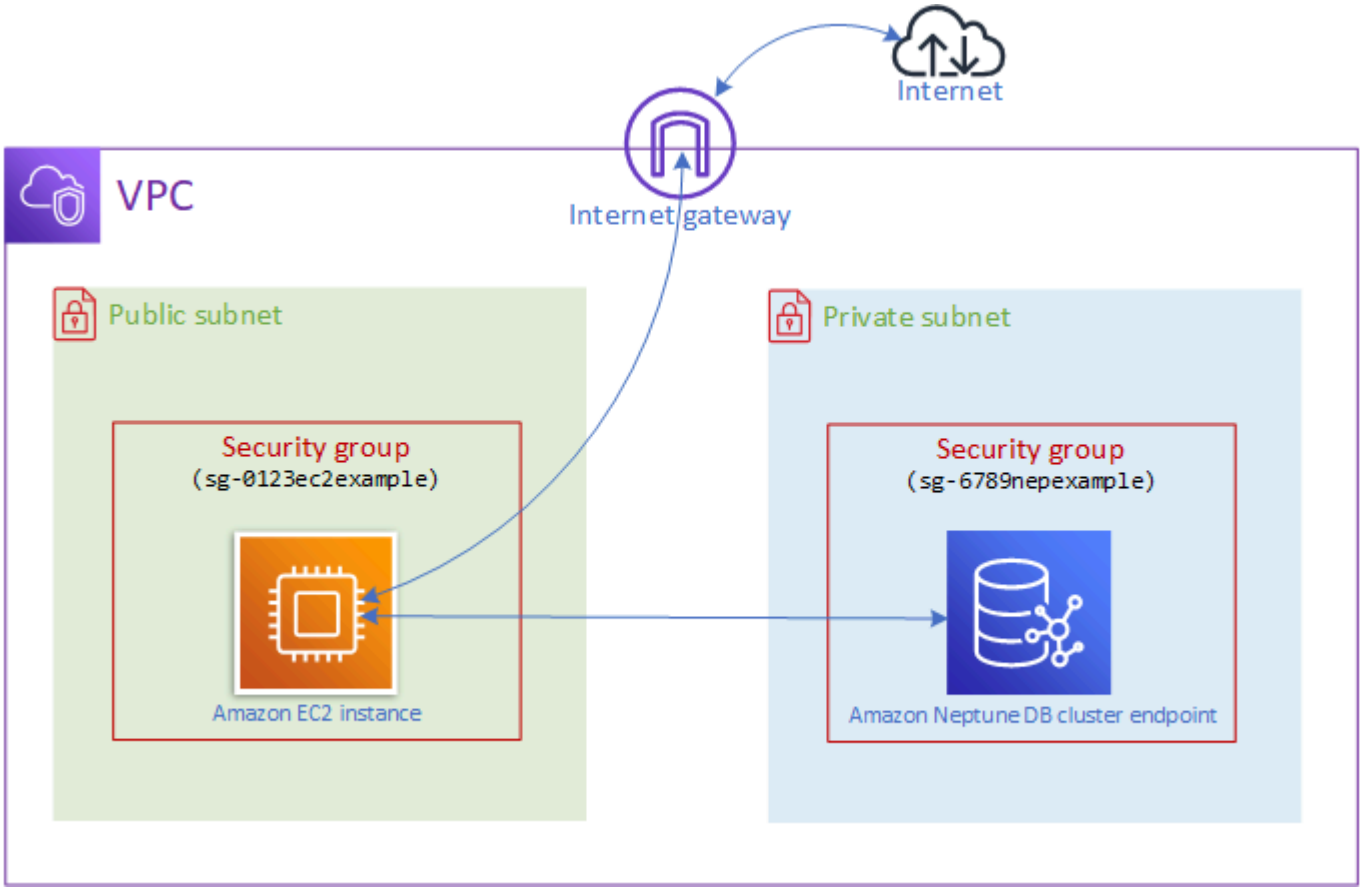
Un cluster di database Amazon Neptune può essere creato solo in un Amazon Virtual Private Cloud (Amazon VPC). A meno che gli endpoint pubblici di Neptune non vengono abilitati e configurati per il cluster di database, i relativi endpoint sono accessibili solo all'interno di tale VPC.

Esistono diversi modi per configurare l'accesso al cluster di database Neptune nel relativo VPC:

- Connessione da un'istanza Amazon EC2 nello stesso VPC
- Connessione da un'istanza Amazon EC2 in un altro VPC
- Connessione da una rete privata

Connessione a un cluster database Neptune da un'istanza Amazon EC2 nello stesso VPC

Uno dei modi più comuni per connettersi a un database Neptune è da un'istanza Amazon EC2 nello stesso VPC del cluster database Neptune. Ad esempio, l'istanza EC2 potrebbe eseguire un server Web che si connette a Internet. In questo caso, solo l'istanza EC2 ha accesso al cluster database Neptune e Internet ha accesso solo all'istanza EC2:



Per abilitare questa configurazione, è necessario che siano configurati i gruppi di sicurezza VPC e i gruppi di sottorete corretti. Il server Web è ospitato in una sottorete pubblica, in modo da poter raggiungere la rete Internet pubblica, mentre l'istanza del cluster Neptune è ospitata in una sottorete privata per garantirne la sicurezza. Per informazioni, consulta Configurare l'Amazon VPC in cui si trova il cluster database Amazon Neptune.

Affinché l'istanza Amazon EC2 possa connettersi all'endpoint Neptune, ad esempio, sulla porta 8182, sarà necessario impostare un gruppo di sicurezza per farlo. Se l'istanza Amazon EC2 utilizza un gruppo di sicurezza denominato, ad esempio, `ec2-sg1`, è necessario creare un altro gruppo di sicurezza Amazon EC2, ad esempio `db-sg1`, che abbia regole in ingresso per la porta `8182` e che abbia `ec2-sg1` come origine. Quindi, aggiungere `db-sg1` al cluster Neptune per consentire la connessione.

Dopo aver creato l'istanza Amazon EC2, è possibile accedervi tramite SSH e connettersi al cluster database Neptune. Per informazioni sulla connessione a un'istanza EC2 con SSH, consulta Connessione all'istanza di Linux nella Guida per l'utente di Amazon EC2 per le istanze Linux.

Se utilizzi una riga di comando Linux o macOS per connetterti all'istanza EC2, puoi incollare il comando SSH dall'elemento SSHAccess nella sezione Outputs dello stack. AWS CloudFormation È necessario che il file PEM si trovi nella directory attuale e che le autorizzazioni relative a tale file siano impostate su 400 (`chmod 400 keypair.pem`).

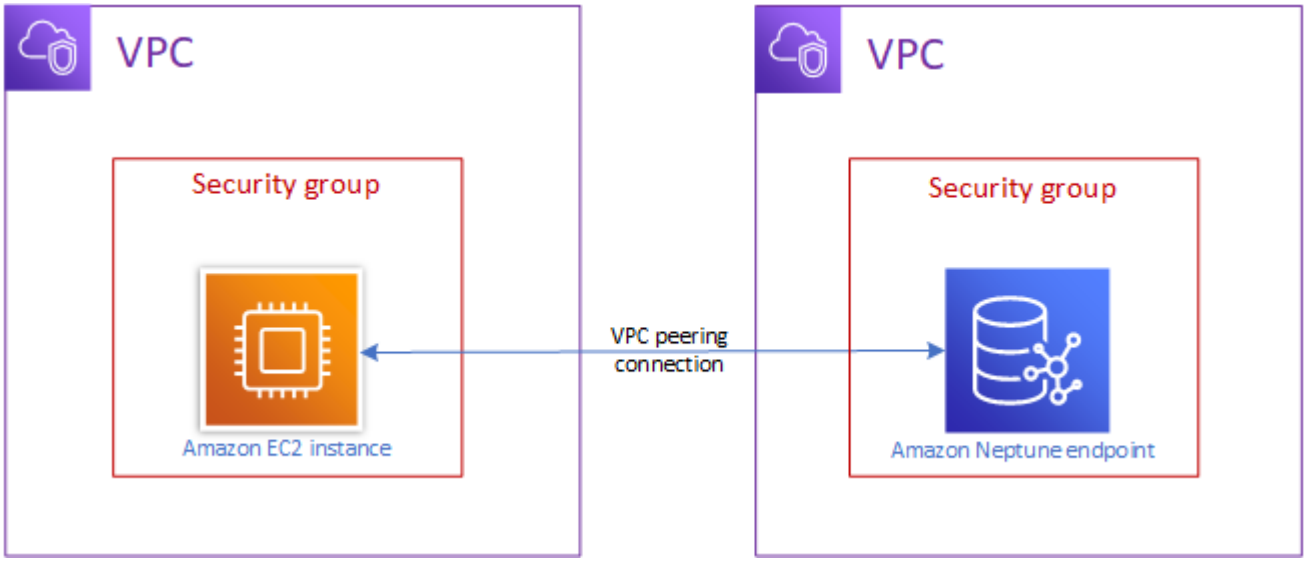
Per creare un VPC con sottoreti pubbliche e private

1. Accedi AWS Management Console e apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nell'angolo in alto a destra di AWS Management Console, scegli la regione in cui creare il tuo VPC.
3. Nel Pannello di controllo VPC, scegliere Avvio della procedura guidata VPC.
4. Completare l'area Impostazioni VPC della pagina Crea VPC:
    a. Alla voce Resources to Create (Risorse da creare), seleziona VPC, subnets, etc. (Sottoreti, VPC e altro).
    b. Lasciare invariato il tag nome predefinito o inserire un nome a scelta oppure deselezionare la casella di controllo Generazione automatica per disabilitare la generazione del tag nome.
    c. Lasciare il valore del blocco CIDR IPv4 su `10.0.0.0/16`.
    d. Lasciare il valore del blocco CIDR IPv6 su Nessun blocco CIDR IPv6.
    e. Lasciare il valore di Tenancy su Default.
    f. Lasciare il numero di Zone di disponibilità (AZ) su 2.
    g. Lasciare il valore di Gateway NAT ($) su Nessuno, a meno che non siano necessari uno o più gateway NAT.
    h. Impostare il valore di Endpoint VPC su Nessuno, a meno che non si utilizzi Amazon S3.
    i. È necessario selezionare sia Abilita nomi host DNS che Abilita risoluzione DNS.
5. Seleziona Crea VPC.

Accesso al cluster database da un'istanza Amazon EC2 in un altro VPC

Un cluster database Amazon Neptune può essere creato solo in un Amazon Virtual Private Cloud (Amazon VPC) e i relativi endpoint sono accessibili solo all'interno di tale VPC, in genere da un'istanza Amazon Elastic Compute Cloud (Amazon EC2) in esecuzione in tale VPC.

Quando il cluster database si trova in un VPC diverso dall'istanza EC2 che si sta utilizzando per accedervi, è possibile utilizzare il peering VPC per effettuare la connessione:



Una connessione peering VPC è una connessione di rete tra due VPC che instrada il traffico tra gli stessi in modo privato, in modo che le istanze in entrambi i VPC possano comunicare come se si trovassero nella stessa rete. Puoi creare una connessione peering VPC tra i VPC del tuo account, tra un VPC nel tuo account AWS e un VPC in un altro account o con un VPC in un'altra regione. AWS AWS

AWS utilizza l'infrastruttura esistente di un VPC per creare una connessione peering VPC. Non è né un gateway né una connessione AWS VPN da sito a sito e non si basa su un hardware fisico separato. Non presenta un singolo punto di errore per la comunicazione né colli di bottiglia nella larghezza di banda.

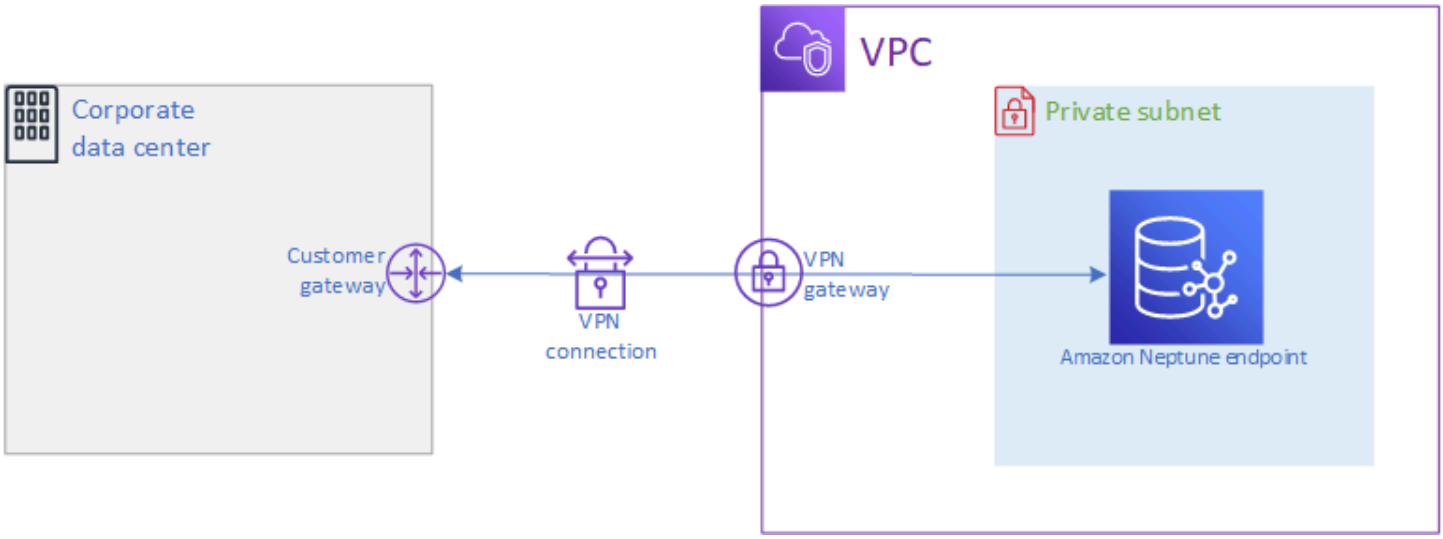
Per ulteriori informazioni su come utilizzare il peering VPC, consulta la Guida di Amazon VPC Peering.

Accesso al cluster database da una rete privata

È possibile accedere a un cluster database Neptune da una rete privata in due modi diversi:

- Usando una connessione VPN sito-sito AWS.
- Usando una connessione Direct Connect AWS.

I collegamenti sopra riportati contengono informazioni su questi metodi di connessione e su come configurarli. La configurazione di una connessione da sito a AWS sito potrebbe essere simile alla seguente:



Protezione dei dati in Amazon Neptune

Esistono diversi modi per proteggere i cluster Amazon Neptune.

Utilizzo delle policy IAM per limitare l'accesso a un cluster database Neptune

Per controllare chi può eseguire azioni di gestione di Neptune su cluster e istanze DB Neptune, usa (IAM). AWS Identity and Access Management

Quando utilizzi un account IAM per accedere alla console Neptune, devi prima accedere utilizzando AWS Management Console il tuo account IAM prima di aprire la console Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.

Quando ti connetti AWS utilizzando le credenziali IAM, il tuo account IAM deve disporre di policy IAM che concedano le autorizzazioni necessarie per eseguire le operazioni di gestione di Neptune. Per ulteriori informazioni, consulta Utilizzo di diversi tipi di policy IAM per controllare l'accesso a Neptune.

Utilizzo di gruppi di sicurezza VPC per limitare l'accesso a un cluster database Neptune

I cluster database Neptune devono essere creati in un Amazon Virtual Private Cloud (Amazon VPC). Per controllare i dispositivi e le istanze EC2 che possono aprire le connessioni all'endpoint e alla

porta dell'istanza database per i cluster database Neptune in un VPC, è necessario utilizzare un gruppo di sicurezza VPC. Per ulteriori informazioni sulla funzionalità VPCs, consultare Creare un gruppo di sicurezza tramite la console VPC.

Utilizzo dell'autenticazione IAM per limitare l'accesso a un cluster database Neptune

Se abiliti l'autenticazione AWS Identity and Access Management (IAM) in un cluster Neptune DB, chiunque acceda al cluster DB deve prima essere autenticato. Per informazioni sulla configurazione dell'autenticazione IAM, consulta Panoramica di AWS Identity and Access Management (IAM) in Amazon Neptune.

Per informazioni sull'utilizzo di credenziali temporanee per l'autenticazione, inclusi esempi per Amazon EC2 AWS CLI AWS Lambda, consulta. the section called “Credenziali temporanee”

I seguenti collegamenti forniscono informazioni aggiuntive sulla connessione a Neptune utilizzando l'autenticazione IAM con i singoli linguaggi di query:

Utilizzo di Gremlin con l'autenticazione IAM

- the section called “Console Gremlin”
- the section called “Gremlin Java”
- the section called “Esempio di Python”

> **Note**
>
> Questo esempio si applica a Gremlin e SPARQL.

Utilizzo di openCypher con l'autenticazione IAM

- the section called “Console Gremlin”
- the section called “Gremlin Java”
- the section called “Esempio di Python”

> **Note**
>
> Questo esempio si applica a Gremlin e SPARQL.

Utilizzo di SPARQL con l'autenticazione IAM

- the section called “SPARQL Java (RDF4J e Jena)”
- the section called “Esempio di Python”

> **Note**
>
> Questo esempio si applica a Gremlin e SPARQL.

Nozioni di base sull'accesso al grafo di Neptune

Dopo aver creato un cluster database Neptune e aver configurato la connessione ad esso, il passaggio successivo consiste nel comunicare con esso in modo da caricare dati, effettuare query e così via. Per fare ciò, la maggior parte delle persone utilizza gli strumenti a riga di comando `curl` o `awscurl`.

Configurazione di **`curl`** per la comunicazione con l'endpoint Neptune

Come illustrato in numerosi esempi di questa documentazione, lo strumento a riga di comando curl è un'opzione utile per comunicare con l'endpoint Neptune. Per ulteriori informazioni sullo strumento, consulta la pagina principale curl e scarica il libro Everything curl.

Per eseguire la connessione tramite HTTPS (come consigliato e come Neptune richiede nella maggior parte delle regioni), `curl` ha bisogno dell'accesso a certificati adeguati. Per scoprire come ottenere questi certificati e come formattarli correttamente in un archivio di certificati CA (Certificate Authority) che `curl` può utilizzare, consulta Verifica certificati SSL nella documentazione di `curl`.

È possibile specificare il percorso di questo archivio di certificati CA utilizzando la variabile di ambiente `CURL_CA_BUNDLE`. In Windows, `curl` cerca automaticamente un file denominato `curl-ca-bundle.crt`. Cerca prima nella stessa cartella di `curl.exe`, quindi in altri punti del percorso. Per ulteriori informazioni, consulta Verifica certificati SSL.

Se `curl` è in grado di individuare i certificati adeguati, gestisce le connessioni HTTPS in modo analogo alle connessioni HTTP, senza parametri aggiuntivi. Gli esempi di questa documentazione si basano su questo scenario.

Utilizzo di un linguaggio di query per accedere ai dati del grafo nel cluster database Neptune

Una volta stabilita la connessione, è possibile utilizzare i linguaggi di query Gremlin e openCypher per creare ed eseguire query su un grafo di proprietà oppure il linguaggio di query SPARQL per creare ed eseguire query su un grafo contenente dati RDF.

Linguaggi di query a grafo supportati da Neptune

- Gremlin è un linguaggio di attraversamento del grafo per grafi di proprietà. Una query in Gremlin è un attraversamento composto da passaggi discreti, ognuno dei quali segue un arco fino a un nodo. Per ulteriori informazioni, consulta la documentazione di Gremlin su Apache 3 TinkerPop.

  L'implementazione di Gremlin in Neptune differisce da altre implementazioni, soprattutto se utilizzi Gremlin-Groovy (query Gremlin inviate come testo serializzato). Per ulteriori informazioni, consulta Conformità agli standard Gremlin in Amazon Neptune.
- openCypher è un linguaggio di query dichiarativo per grafi di proprietà che è stato sviluppato originariamente da Neo4j, per poi diventare open source nel 2015, e che ha contribuito al progetto openCypher con una licenza open source Apache 2. La sintassi di openCypher è documentata in Cypher Query Language Reference, versione 9.
- SPARQL è un linguaggio di query dichiarativo per i dati RDF basato sul tipo di corrispondenza del modello di grafo standardizzato dal World Wide Web Consortium (W3C) e descritto in SPARQL 1.1 Overview e nella specifica SPARQL 1.1 Query Language.

> **Note**
>
> È possibile accedere ai dati dei grafi di proprietà in Neptune utilizzando sia Gremlin che openCypher, ma non utilizzando SPARQL. Allo stesso modo, è possibile accedere ai dati RDF solo utilizzando SPARQL, non Gremlin o openCypher.

Utilizzo di Gremlin per accedere al grafo in Amazon Neptune

Puoi usare la Gremlin Console per sperimentare TinkerPop grafici e interrogazioni in un ambiente REPL (loop). read-eval-print

I seguenti tutorial descrivono come utilizzare la console di Gremlin per aggiungere vertici, edge, proprietà e altro a un grafo Neptune, oltre a evidenziare alcune differenze nell'implementazione di Gremlin specifica per Neptune.

> **Note**
>
> Questo esempio presuppone che siano state completate le seguenti operazioni:
>
> - Sei connesso a un'istanza Amazon EC2 tramite SSH.
> - Hai creato un cluster Neptune come descritto in Creazione di un cluster DB.
> - Hai installato la console di Gremlin come descritto in Installazione della console Gremlin.

Utilizzi la console di Gremlin

1. Cambia directory nelle cartelle in cui vengono decompressi i file della console di Gremlin.

```
cd apache-tinkerpop-gremlin-console-3.6.5
```

2. Digita il comando seguente per eseguire la console di Gremlin.

```
bin/gremlin.sh
```

   Verrà visualizzato l'output seguente:

```
         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin>
```

   Ora ti trovi al prompt `gremlin>`. Immetti i restanti passaggi in questo prompt.

3. Al prompt `gremlin>`, immetti quanto segue per connetterti all'istanza database Neptune.

```
:remote connect tinkerpop.server conf/neptune-remote.yaml
```

4. Al prompt `gremlin>`, immetti quanto segue per passare alla modalità remota. In questo modo tutte le query Gremlin vengono inviate alla connessione remota.

   ```
   :remote console
   ```

5. Aggiungi un vertice con etichetta e proprietà.

   ```
   g.addV('person').property('name', 'justin')
   ```

   Al vertice viene assegnato un ID `string` contenente un GUID. Tutti gli ID dei vertici sono stringhe presenti in Neptune.

6. Aggiungi un vertice con ID personalizzato.

   ```
   g.addV('person').property(id, '1').property('name', 'martin')
   ```

   La proprietà `id` non è indicata. Si tratta di una parola chiave per l'ID del vertice. L'ID del vertice qui indicato è una stringa contenente il numero `1`.

   I nomi di proprietà standard devono essere racchiusi tra virgolette.

7. Cambia la proprietà o aggiungila se non è già stata creata.

   ```
   g.V('1').property(single, 'name', 'marko')
   ```

   Qui è possibile modificare la proprietà `name` per il vertice della fase precedente. Questo consente di rimuovere tutti i valori esistenti della proprietà `name`.

   Se non hai specificato `single`, il valore viene aggiunto alla proprietà `name` (se ancora non è stato fatto).

8. Aggiungi la proprietà, ma aggiungila in coda se dispone già di un valore.

   ```
   g.V('1').property('age', 29)
   ```

   Neptune impiega la cardinalità di un insieme come azione predefinita.

   Questo comando aggiunge la proprietà `age` con il valore `29`, ma non sostituisce i valori esistenti.

   Se la proprietà `age` disponeva già di un valore, questo comando aggiunge `29` in coda alla proprietà. Ad esempio, se la proprietà `age` era `27`, il nuovo valore è `[ 27, 29 ]`.

9. Aggiungi più vertici.

```
g.addV('person').property(id, '2').property('name', 'vadas').property('age',
 27).iterate()
g.addV('software').property(id, '3').property('name', 'lop').property('lang',
 'java').iterate()
g.addV('person').property(id, '4').property('name', 'josh').property('age',
 32).iterate()
g.addV('software').property(id, '5').property('name', 'ripple').property('lang',
 'java').iterate()
g.addV('person').property(id, '6').property('name', 'peter').property('age', 35)
```

È possibile inviare più istruzioni contemporaneamente a Neptune.

Le istruzioni possono essere separate da nuove righe (`'\n'`), spazi (`' '`), punto e virgola (`'; '`) o non essere separate da alcun carattere (ad esempio, l'istruzione `g.addV('person').iterate()g.V()` è valida).

> **Note**
>
> La console di Gremlin invia un comando separato a ogni nuova riga (`'\n'`), di modo che risulteranno essere transazioni distinte. Questo esempio mostra tutti i comandi su righe separate per motivi di leggibilità. Rimuovi i caratteri della nuova riga (`'\n'`) per inviarli in un unico comando tramite la console Gremlin.

Tutte le istruzioni diverse dall'ultima devono avere una fase di chiusura, ad esempio `.next()` o `.iterate()`; in caso contrario non saranno eseguite. La console di Gremlin non richiede queste fasi finali. Usa `.iterate` quando non è necessario che i risultati vengano serializzati.

Tutte le istruzioni che vengono inviate insieme sono incluse in una singola transazione e insieme hanno esito positivo o negativo.

10. Aggiungi edge.

```
g.V('1').addE('knows').to(__.V('2')).property('weight', 0.5).iterate()
g.addE('knows').from(__.V('1')).to(__.V('4')).property('weight', 1.0)
```

Di seguito sono indicati due diversi modi per aggiungere un edge.

11. Aggiungi il resto del grafo moderno.

```
g.V('1').addE('created').to(__.V('3')).property('weight', 0.4).iterate()
g.V('4').addE('created').to(__.V('5')).property('weight', 1.0).iterate()
g.V('4').addE('knows').to(__.V('3')).property('weight', 0.4).iterate()
g.V('6').addE('created').to(__.V('3')).property('weight', 0.2)
```

12. Elimina un vertice.

```
g.V().has('name', 'justin').drop()
```

Rimuove il vertice con la proprietà `name` uguale a `justin`.

> **Important**
>
> Fermatevi qui e avrete il grafico completo di Apache Modern. TinkerPop Gli esempi nella sezione Traversal della TinkerPop documentazione utilizzano il grafico Modern.

13. Esegui un attraversamento.

```
g.V().hasLabel('person')
```

Restituisce tutti i vertici `person`.

14. Esegui un attraversamento con valori (valueMap ()).

```
g.V().has('name', 'marko').out('knows').valueMap()
```

Restituisce coppie di chiavi e di valori per tutti i vertici riconosciuti da `marko`.

15. Specifica più etichette.

```
g.addV("Label1::Label2::Label3")
```

Neptune supporta più etichette per un vertice. Quando crei un'etichetta, puoi specificare più etichette separandole con `::`.

In questo esempio viene aggiunto un vertice con tre etichette diverse.

La fase `hasLabel` corrisponde a questo vertice con una qualsiasi delle tre etichette: `hasLabel("Label1")` , `hasLabel("Label2")` e `hasLabel("Label3")`.

Il delimitatore `::` è riservato solo a quest'uso.

Non è possibile specificare più etichette nella fase `hasLabel`. Ad esempio, `hasLabel("Label1::Label2")` non corrisponde a nulla.

16. Specifica Ora/data.

```
g.V().property(single, 'lastUpdate', datetime('2018-01-01T00:00:00'))
```

Neptune non supporta la data Java. Utilizza invece la funzione `datetime()`. `datetime()` accetta una stringa `datetime` conforme allo standard ISO8061.

Supporta i seguenti formati: `YYYY-MM-DD`, `YYYY-MM-DDTHH:mm`, `YYYY-MM-DDTHH:mm:SS` e `YYYY-MM-DDTHH:mm:SSZ`.

17. Elimina vertici, proprietà o edge.

```
g.V().hasLabel('person').properties('age').drop().iterate()
g.V('1').drop().iterate()
g.V().outE().hasLabel('created').drop()
```

Di seguito sono riportati vari esempi.

> **Note**
>
> La fase `.next()` non funziona con `.drop()`. Usare invece `.iterate()`.

18. Al termine, immetti quanto segue per uscire dalla console di Gremlin.

```
:exit
```

> **Note**
>
> Utilizzare un punto e virgola (`;`) o un carattere nuova riga (`\n`) per separare ogni istruzione. Ogni attraversamento che precede l'attraversamento finale deve terminare in `iterate()` per essere eseguito. Vengono restituiti solo i dati dell'ultimo attraversamento.

Utilizzo di openCypher per accedere al grafo in Amazon Neptune

Per iniziare a usare OpenCypher, consulta o usa i taccuini openCypher OpenCypher nel repository graf-notebook Neptune. GitHub

Utilizzo di RDF e SPARQL per l'accesso al grafo in Amazon Neptune

SPARQL è un linguaggio di query per l'RDF (Resource Description Framework), un formato di dati a grafo progettato per il Web. Amazon Neptune è compatibile con SPARQL 1.1. Questo ti consente di connetterti a un'istanza database Neptune ed eseguire query sul grafo utilizzando il linguaggio di query descritto nella specifica SPARQL 1.1 Query Language.

Una query in SPARQL consiste di una clausola `SELECT` per specificare le variabili da restituire e una clausola `WHERE` per specificare i dati da abbinare nel grafo. Per ulteriori informazioni sulle query SPARQL, vedere Writing Simple Queries (Scrittura di query semplici) in SPARQL 1.1 Query Language.

L'endpoint HTTP per le query SPARQL in un'istanza database Neptune è `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.

Connessione a SPARQL

1. Puoi ottenere l'endpoint SPARQL per il tuo cluster Neptune dall'SparqlEndpointelemento nella sezione Outputs dello stack. AWS CloudFormation
2. Digita quanto segue per inviare un **`UPDATE`** SPARQL utilizzando il `POST` HTTP e il comando curl.

   ```
   curl -X POST --data-binary 'update=INSERT DATA { <https://test.com/s> <https://
   test.com/p> <https://test.com/o> . }' https://your-neptune-endpoint:port/sparql
   ```

   L'esempio precedente inserisce la seguente tripla nel grafo SPARQL predefinito: `<https://test.com/s> <https://test.com/p> <https://test.com/o>`
3. Digita quanto segue per inviare un **`QUERY`** SPARQL utilizzando il `POST` HTTP e il comando curl.

   ```
   curl -X POST --data-binary 'query=select ?s ?p ?o where {?s ?p ?o} limit 10'
    https://your-neptune-endpoint:port/sparql
   ```

   L'esempio precedente restituisce fino a 10 delle triple (subject-predicate-object) nel grafico utilizzando la query con un limite di 10. `?s ?p ?o` Per eseguire una query su qualcos'altro, sostituirla con un'altra query SPARQL .

> **Note**
>
> Il tipo MIME predefinito di una risposta è `application/sparql-results+json` per le query `SELECT` e `ASK`.
> Il tipo MIME predefinito di una risposta è `application/n-quads` per le query `CONSTRUCT` e `DESCRIBE`.
> Per un elenco di tutti i tipi MIME disponibili, vedi API SPARQL HTTP.

Caricamento di dati in Neptune

In Amazon Neptune è disponibile un processo che consente di caricare dati da file esterni direttamente in un'istanza database Neptune. Puoi utilizzare questo processo come alternativa all'esecuzione di un numero elevato di istruzioni `INSERT`, fasi `addV` e `addE` o altre chiamate API.

Di seguito sono elencati i collegamenti di ulteriori informazioni sul caricamento.

- Metodi per il caricamento dei dati – Caricamento dei dati
- Formati di dati supportati dallo strumento di caricamento in blocco – the section called “Formati dei dati”
- Esempio di caricamento – the section called “Esempio di caricamento”

Monitoraggio di Amazon Neptune

Amazon Neptune supporta i seguenti metodi di monitoraggio.

- Amazon CloudWatch — Amazon Neptune invia automaticamente i parametri e supporta anche gli CloudWatch allarmi. CloudWatch Per ulteriori informazioni, consulta the section called “Usando CloudWatch”.
- AWS CloudTrail— Amazon Neptune supporta la registrazione delle API utilizzando. CloudTrail Per ulteriori informazioni, consulta the section called “Registrazione delle chiamate API Neptune con AWS CloudTrail”.
- Tagging: usa i tag per aggiungere metadati alle risorse Neptune e monitorare l'utilizzo in base a tag. Per ulteriori informazioni, consulta the section called “Applicazione di tag alle risorse Neptune”.

- File di log di audit: puoi visualizzare, scaricare o controllare i file di log del database tramite la console. Per ulteriori informazioni, consulta the section called “Log di audit con Neptune”.
- Stato dell'istanza: controlla lo stato del motore di database a grafo di un'istanza Neptune, scopri quale versione del motore è installata e ottieni altre informazioni sullo stato del motore utilizzando l'API dello stato dell'istanza.

Risoluzione dei problemi e best practice in Neptune

I seguenti collegamenti possono essere utili per la risoluzione dei problemi di Amazon Neptune.

- Best practice: per le soluzioni a problemi comuni e i suggerimenti sulle prestazioni, vedi Best practice.
- Errori del servizio: per un elenco di errori per le API di gestione e per le connessioni del database a grafo, vedi Errori Neptune.
- Limiti del servizio: per informazioni sui limiti di Neptune, vedi Limiti di Neptune.
- Versioni del motore: per informazioni sui rilasci del motore a grafo, incluse le note di rilascio, vedi Rilasci del motore.
- Forum di supporto: per partecipare a una discussione su Neptune, vedi il Forum di Amazon Neptune.
- Prezzi: per informazioni sui costi di utilizzo di Amazon Neptune, vedi Prezzi di Amazon Neptune.
- AWS Supporto: per l'aiuto e la guida degli esperti, consulta AWS Support.

Creazione di un database globale Neptune

Un database globale Amazon Neptune si estende su più Regioni AWS, consentendo letture globali a bassa latenza e fornendo un ripristino rapido nel raro caso in cui un'interruzione interessi un'intera Regione AWS.

Un database globale Neptune è costituito da un cluster database primario in una regione e fino a cinque cluster database secondari in regioni diverse.

Le scritture possono avvenire solo nella regione primaria. Le regioni secondarie supportano solo le letture. In ogni regione secondaria possono essere presenti fino a 16 istanze di lettura.

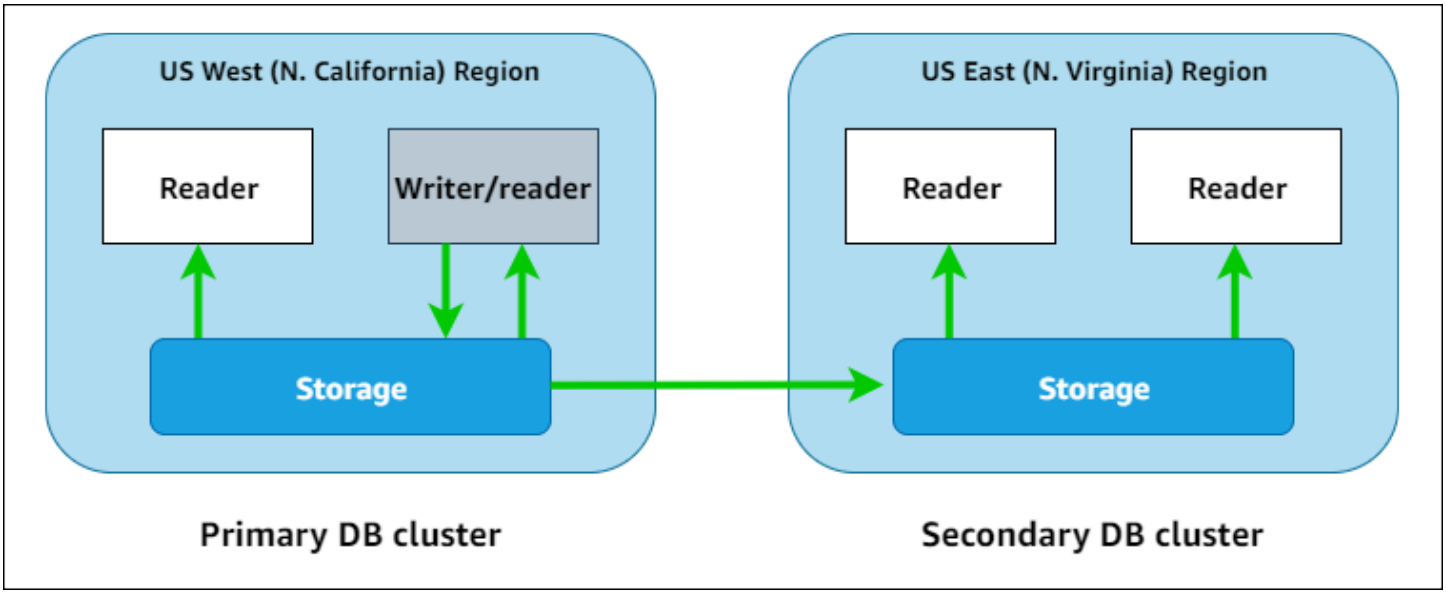
Argomenti

Panoramica dei database globali Amazon Neptune

Utilizzando un database globale Neptune, puoi eseguire le applicazioni distribuite a livello globale in un singolo database che si estende su più Regioni AWS.

Un database globale Neptune è costituito da un cluster database primario in una Regione AWS primaria in cui vengono scritti i dati e fino a cinque cluster database di sola lettura in una Regioni AWS secondaria. Quando si esegue un'operazione di scrittura sul cluster database primario, Neptune replica i dati scritti su tutti i cluster database secondari tramite un'infrastruttura dedicata, con una latenza in genere inferiore a un secondo.

Nel diagramma seguente viene illustrato un esempio di database globale che si estende su due Regioni AWS:



Puoi dimensionare ogni cluster secondario in modo indipendente per gestire i carichi di sola lettura aggiungendo una o più istanze di replica di lettura.

Per eseguire operazioni di scrittura è necessario connettersi all'endpoint del cluster database del cluster database primario. Solo il cluster primario eseguire operazioni di scrittura. Quindi, come mostrato nel diagramma precedente, la replica viene eseguita dal volume di archiviazione del cluster, non dal motore del database.

I database globali Neptune sono progettati per le applicazioni con una presenza globale. I cluster database secondari di sola lettura supportano operazioni di lettura più vicine agli utenti delle applicazioni.

Un database globale Neptune supporta due diversi approcci al failover.

- Per eseguire il ripristino da un'interruzione nella regione primaria, utilizza il processo di scollegamento e promozione manuali non pianificati, in cui uno dei cluster secondari viene scollegato, trasformato in un cluster autonomo e quindi promosso come nuovo cluster primario.
- Per le procedure operative pianificate come la manutenzione, è possibile utilizzare il failover pianificato gestito, in cui si trasferisce il cluster primario in una delle regioni secondarie senza alcuna perdita di dati.

Vantaggi dell'utilizzo dei database globali in Amazon Neptune

L'utilizzo di database globali offre i seguenti vantaggi:

- Letture globali con latenza locale: se l'azienda ha uffici in tutto il mondo, l'utilizzo di un database globale consente agli uffici nelle regioni secondarie di accedere ai dati nella propria regione con latenza locale.
- Cluster di database Neptune secondari scalabili: è possibile aumentare le dimensioni dei cluster secondari aggiungendo più istanze database di replica di lettura. Poiché i cluster secondari sono di sola lettura, ognuno di essi può supportare fino a 16 repliche di lettura, anziché il limite abituale di 15.
- Replica rapida su cluster database secondari: la replica da cluster database primari a secondari è rapida, con latenza in genere inferiore a un secondo, con un impatto ridotto sulle prestazioni del cluster database primario. Poiché la replica viene eseguita a livello di archiviazione, le risorse delle istanze database sono completamente disponibili per i carichi di lavoro di lettura e scrittura delle applicazioni.
- Ripristino da interruzioni a livello regionale: i cluster database secondari consentono di spostare il cluster primario in una nuova regione più rapidamente, con un RTO minore e con una perdita di dati inferiore (RPO minore) rispetto alle soluzioni di replica tradizionali.

Limitazioni dei database globali in Amazon Neptune

Le seguenti limitazioni si applicano attualmente agli Global Database:

- I database globali Neptune sono disponibili solo nelle Regioni AWS seguenti:
  - Stati Uniti orientali (Virginia settentrionale): `us-east-1`
  - Stati Uniti orientali (Ohio): `us-east-2`
  - Stati Uniti occidentali (California settentrionale): `us-west-1`
  - Stati Uniti occidentali (Oregon): `us-west-2`
  - Europa (Irlanda): `eu-west-1`
  - Europa (Londra): `eu-west-2`
  - Asia Pacifico (Tokyo): `ap-northeast-1`
- I database globali Neptune non supportano il dimensionamento automatico per i cluster database secondari.
- Non è possibile applicare un gruppo di parametri personalizzato al cluster database globale mentre si esegue un aggiornamento della versione principale del database globale. Prova a creare i gruppi di parametri personalizzati in ogni regione del cluster globale e quindi applicali manualmente ai cluster regionali dopo l'aggiornamento.

- Non è possibile interrompere o avviare singolarmente i cluster database nel database globale.
- Le istanze di replica di lettura in un cluster database secondario possono essere riavviate in determinate circostanze. Se l'istanza di scrittura del cluster primario viene riavviata o esegue il failover, verranno riavviate anche tutte le istanze nelle regioni secondarie. Il cluster secondario non sarà quindi disponibile fino a quando tutte le istanze al suo interno non sono nuovamente sincronizzate con l'istanza di scrittura del cluster database primario.

Configurazione di un database globale in Amazon Neptune

È possibile creare un database globale Neptune in uno dei modi seguenti:

- Creare un database globale con nuovi cluster e istanze database.
- Creare un database globale usando un cluster database Neptune esistente come cluster primario.

Argomenti

Requisiti di configurazione per un database globale in Amazon Neptune

Un database globale Neptune si estende su almeno due Regioni AWS. La Regione AWS primaria contiene un cluster database Neptune con un'istanza di scrittura. Da una a cinque Regioni AWS secondarie, ognuna contenente un cluster database Neptune di sola lettura costituito interamente da istanze di replica di lettura. È richiesta almeno una Regione AWS secondaria.

I cluster database Neptune che costituiscono un database globale prevedono i requisiti specifici seguenti:

- Requisiti delle classi di istanza database: un database globale richiede classi di istanza database `r5` o `r6g` ottimizzate per carichi di lavoro a utilizzo intensivo di memoria, come un tipo di istanza `db.r5.large`.

- Requisiti per Regione AWS: un database globale richiede un cluster database Neptune primario in una Regione AWS e almeno un cluster database Neptune secondario in una regione diversa. È possibile creare fino a cinque cluster database Neptune di sola lettura secondari e ognuno di essi deve risiedere in una regione diversa. In altre parole, due cluster database Neptune in un database globale Neptune non possono trovarsi nella stessa Regione AWS.
- Requisiti della versione del motore: la versione del motore Neptune utilizzata da tutti i cluster database nel database globale deve essere la stessa e deve essere maggiore o uguale alla versione `1.2.0.0`. Se non si specifica la versione del motore durante la creazione di un nuovo database, cluster o istanza globale, verrà utilizzata la versione del motore più recente.

> **Important**
>
> Sebbene i gruppi di parametri del cluster database possano essere configurati indipendentemente per ogni cluster database in un database globale, è consigliabile mantenere le impostazioni coerenti tra i cluster per evitare modifiche di comportamento impreviste nel caso in cui un cluster secondario debba essere promosso a primario. Utilizza ad esempio le stesse impostazioni per gli indici di oggetti, i flussi e così via in tutti i cluster database.

Utilizzo della AWS CLI per creare un database globale in Amazon Neptune

> **Note**
>
> Gli esempi in questa sezione seguono la convenzione UNIX che prevede l'utilizzo di una barra rovesciata (\) come carattere di estensione della linea. Per Windows, sostituisci la barra rovesciata con un accento circonflesso (^)

Per creare un database globale usando la AWS CLI

1. Crea un database globale vuoto utilizzando il comando `create-global-cluster` AWS CLI (che racchiude l'API CreateGlobalCluster). Specificate il nome della Regione AWS primaria, imposta Neptune come motore di database e, facoltativamente, specifica la versione del motore da utilizzare (deve essere la versione 1.2.0.0 o successiva):

   ```
   aws neptune create-global-cluster
   ```

```
  --region (primary region, such as us-east-1) \
  --global-cluster-identifier (ID for the global database) \
  --engine neptune \
  --engine-version (engine version; this is optional)
```

2. Potrebbero essere necessari alcuni minuti prima che il database globale sia disponibile, quindi prima di procedere al passaggio successivo, usa il comando `describe-global-clusters` della CLI (che racchiude l'API DescribeGlobalClusters) per verificare che il database globale sia disponibile:

```
aws neptune describe-global-clusters \
  --region (primary region) \
  --global-cluster-identifier (global database ID)
```

3. Non appena il database globale Neptune è disponibile, puoi creare un nuovo cluster database Neptune come cluster primario:

```
aws neptune create-db-cluster \
  --region (primary region) \
  --db-cluster-identifier (ID for the primary DB cluster) \
  --engine neptune \
  --engine-version (engine version; must be >= 1.2.0.0) \
  --global-cluster-identifier (global database ID)
```

4. Usa il comando `describe-db-clusters` AWS CLI per confermare che il nuovo cluster database è pronto per l'aggiunta dell'istanza database primaria:

```
aws neptune describe-db-clusters \
  --region (primary region) \
  --db-cluster-identifier (primary DB cluster ID)
```

   Quando la risposta mostra lo stato `"Status": "available"`, procedi al passaggio successivo.

5. Crea l'istanza database primaria per il cluster primario utilizzando il comando `create-db-instance` AWS CLI. È necessario utilizzare uno dei tipi di istanza `r5` o `r6g` ottimizzati per la memoria, ad esempio `db.r5.large`.

```
aws neptune create-db-instance \
  --region (primary region) \
  --db-cluster-identifier (primary cluster ID) \
```

```
  --db-instance-class (instance class) \
  --db-instance-identifier (ID for the DB instance) \
  --engine neptune \
  --engine-version (optional: engine version)
```

> **Note**
>
> Se si prevede di aggiungere dati al nuovo cluster database primario utilizzando lo strumento di caricamento in blocco Neptune, è necessario procedere prima di aggiungere le regioni secondarie. Questa operazione è più rapida ed economica rispetto all'esecuzione di un caricamento in blocco dopo la configurazione completa del database globale.

Ora puoi aggiungere una o più aree secondarie al nuovo database globale, come descritto in Aggiunta di una regione secondaria con la AWS CLI.

Trasformazione di un cluster database esistente in un database globale

Per trasformare un cluster database esistente in un database globale, usa il comando `create-global-cluster` AWS CLI per creare un nuovo database globale nella stessa Regione AWS in cui si risiede il cluster database esistente e imposta il relativo parametro `--source-db-cluster-identifier` sul nome della risorsa Amazon (ARN) del cluster esistente che si trova in quella posizione:

```
aws neptune create-global-cluster \
  --region (region where the existing cluster is located) \
  --global-cluster-identifier (provide an ID for the new global database) \
  --source-db-cluster-identifier (the ARN of the existing DB cluster) \
  --engine neptune \
  --engine-version (engine version; this is optional)
```

Ora puoi aggiungere una o più aree secondarie al nuovo database globale, come descritto in Aggiunta di una regione secondaria con la AWS CLI.

Utilizzo di un cluster database ripristinato da uno snapshot come cluster primario

È possibile trasformare un cluster database ripristinato da uno snapshot in un database globale di Neptune. Una volta completato il ripristino, trasforma il cluster database creato nel cluster primario di un nuovo database globale come descritto sopra.

Aggiunta di regioni di database globale secondarie a una regione primaria in Amazon Neptune

Un database globale Neptune richiede almeno un cluster database Neptune secondario in una regione Regione AWS diversa da quella del cluster database primario. È possibile collegare fino a cinque cluster database secondari al cluster database primario.

Ogni cluster database secondario aggiunto riduce di uno il numero massimo di istanze di replica di lettura consentite sul cluster primario. Ad esempio, se sono presenti quattro cluster secondari, il numero massimo di istanze di replica di lettura consentite sul cluster primario è `15 - 4 = 11`. Ad esempio, se esistono 14 istanze di lettura nel cluster database primario e un cluster secondario, non è possibile aggiungere un altro cluster secondario.

Utilizzo della AWS CLI per aggiungere una regione secondaria a un database globale in Neptune

Per aggiungere una regione secondaria Regione AWS a un database globale Neptune utilizzando la AWS CLI

1. Usa il comando `create-db-cluster` AWS CLI per creare un nuovo cluster database in una regione diversa da quella del cluster primario e imposta il relativo parametro `--global-cluster-identifier` per specificare l'ID del database globale:

   ```
   aws neptune create-db-cluster \
     --region (the secondary region) \
     --db-cluster-identifier (ID for the new secondary DB cluster) \
     --global-cluster-identifier (global database ID)
     --engine neptune \
     --engine-version (optional: engine version)
   ```

2. Usa il comando `describe-db-clusters` AWS CLI per confermare che il nuovo cluster database è pronto per l'aggiunta dell'istanza database primaria:

   ```
   aws neptune describe-db-clusters \
     --region (primary region) \
     --db-cluster-identifier (primary DB cluster ID)
   ```

   Quando la risposta mostra lo stato `"Status": "available"`, procedi al passaggio successivo.

3. Crea l'istanza database primaria per il cluster primario utilizzando il comando `create-db-instance` AWS CLI, usando un tipo di istanza nella classe dell'istanza `r5` o `r6g`:

```
aws neptune create-db-instance \
  --region (secondary region) \
  --db-cluster-identifier (secondary cluster ID) \
  --db-instance-class (instance class) \
  --db-instance-identifier (ID for the DB instance) \
  --engine neptune \
  --engine-version (optional: engine version)
```

> **Note**
>
> Se non intendi gestire un gran numero di richieste di lettura nell'area secondaria e desideri principalmente usarla per la conservazione affidabile dei backup dei dati, puoi creare un cluster database secondario senza istanze database. In questo modo potrai risparmiare denaro, in quanto pagherai solo lo spazio di archiviazione del cluster secondario, che Neptune manterrà sincronizzato con lo spazio di archiviazione nel cluster database primario.

Connessione a un database globale Neptune

La modalità di connessione a un database globale Neptune dipende dall'operazione da eseguire su di esso, ovvero scrittura o lettura.

- Per richieste o query di sola lettura, è necessario connettersi all'endpoint di lettura per il cluster Neptune nella regione Regione AWS.
- Per eseguire query di mutazione, occorre connettersi all'endpoint del cluster per il cluster database primario, che potrebbe trovarsi in una regione Regione AWS diversa rispetto a quella dell'applicazione.

Gestione di un database globale Amazon Neptune

Ad eccezione del processo di failover pianificato gestito, è possibile eseguire la maggior parte delle operazioni di gestione sui singoli cluster che costituiscono un database globale Neptune. Il processo di failover pianificato gestito è disponibile solo per i database globali Neptune, non per i singoli cluster

database Neptune. Per ulteriori informazioni, consulta Esecuzione di failover pianificati gestiti per database globali Neptune.

Per ripristinare un database globale Neptune da un'interruzione non pianificata nella relativa regione primaria, consulta Scollegamento e promozione di un database globale Neptune in caso di interruzione non pianificata.

Sebbene sia possibile configurare i gruppi di parametri del cluster database in modo indipendente per ogni cluster Neptune in un database globale, è consigliabile mantenere le impostazioni coerenti tra tutti i cluster per evitare modifiche di comportamento impreviste se un cluster secondario viene promosso a primario. Utilizza ad esempio le stesse impostazioni per gli indici di oggetti, i flussi e così via in tutti i cluster database.

Rimozione di un cluster da un database globale

Esistono vari motivi per cui potrebbe essere necessario rimuovere un cluster database da un database globale. Ad esempio:

- Se il cluster primario diventa degradato o isolato, è possibile rimuoverlo dal database globale e diventa un cluster autonomo di cui è stato effettuato il provisioning che può essere utilizzato per creare un nuovo database globale (consulta Scollegamento e promozione di un database globale Neptune in caso di interruzione non pianificata).
- Per eliminare un database globale, per prima cosa è necessario rimuovere (scollegare) tutti i cluster associati da esso, lasciando per ultimo quello primario (consulta Eliminazione di un database globale Neptune)

Puoi usare il comando CLI remove-from-global-cluster (che racchiude l'API RemoveFromGlobalCluster) per scollegare un cluster database Neptune da un database globale:

```
aws neptune remove-from-global-cluster \
  --region (region of the cluster to remove) \
  --global-cluster-identifier (global database ID) \
  --db-cluster-identifier (ARN of the cluster to remove)
```

Il cluster database scollegato diventa quindi un cluster database autonomo.

Eliminazione di un database globale Neptune

Non è possibile eliminare un database globale e i cluster associati in un singolo passaggio. È invece necessario eliminare i suoi componenti uno per uno:

1. Scollega tutti i cluster database secondari dal database globale, come descritto in Rimozione di un cluster. Se lo desideri, ora puoi eliminarli singolarmente.
2. Rimuovi il cluster database primario dal database globale.
3. Elimina tutte le istanze database di replica di lettura dal cluster primario.
4. Elimina l'istanza database (scrittura) primaria dal cluster primario. Se esegue questa operazione nella console, viene eliminato anche il cluster database.
5. Elimina il database globale stesso. Per eseguire questa operazione nella AWS CLI, usa il comando CLI delete-global-cluster (che racchiude l'API DeleteGlobalCluster), nel modo seguente:

```
aws neptune delete-global-cluster \
  --region (region of the DB cluster to delete) \
  --global-cluster-identifier (global database ID)
```

Modifica di un database globale Neptune

I gruppi di parametri del cluster database possono essere configurati in modo indipendente per ogni cluster database Neptune in un database globale, ma è consigliabile mantenere le impostazioni coerenti tra i cluster per evitare modifiche di comportamento impreviste se un cluster secondario deve essere promosso a primario.

È possibile modificare le impostazioni del database globale stesso utilizzando il comando CLI modify-global-cluster (che racchiude l'API ModifyGlobalCluster). Ad esempio, è possibile modificare l'ID del database globale e allo stesso tempo disattivare la protezione dall'eliminazione in questo modo:

```
aws neptune modify-global-cluster \
  --region (region of the DB cluster to modify) \
  --global-cluster-identifier (current global database ID) \
  --new-global-cluster-identifier (new global database ID to assign) \
  --deletion-protection false
```

Utilizzo del failover in un database globale Neptune

Un database globale Neptune offre funzionalità di failover più complete rispetto a un cluster database Neptune autonomo. Con un database globale, è possibile pianificare ed eseguire il ripristino da un'emergenza abbastanza rapidamente. Il ripristino di emergenza viene generalmente determinato tramite la valutazione dell'Obiettivo del tempo di ripristino (RTO) e dell'Obiettivo del punto di ripristino (RPO):

- Obiettivo del tempo di ripristino (RTO): indica la velocità con cui un sistema torna a uno stato operativo dopo un'emergenza. In altre parole, l'RTO misura i tempi di inattività. Per un database globale Neptune, l'RTO può essere nell'ordine di minuti.
- Obiettivo del punto di ripristino (RPO): quantità di tempo durante la quale i dati sono persi. Per un database globale Neptune, l'RPO viene in genere misurato in secondi (consulta Esecuzione di failover pianificati gestiti per database globali Neptune).

Per un database globale Neptune sono disponibili due diversi approcci al failover:

- Scollegamento e promozione (ripristino manuale non pianificato): per eseguire il ripristino da un'interruzione non pianificata o per eseguire un test di ripristino di emergenza, puoi eseguire un processo di scollegamento e promozione tra regioni in uno dei cluster database secondari nel database globale. L'RTO per questo processo manuale dipende dalla rapidità con cui è possibile eseguire le attività elencate in Scollegamento e promozione. L'RPO viene in genere misurato in pochi secondi, ma dipende dal ritardo della replica di archiviazione sulla rete al momento dell'errore.
- Failover pianificato gestito: questo approccio è destinato alla manutenzione operativa e ad altre procedure operative pianificate, come il trasferimento del cluster database primario del database globale in una delle regioni secondarie. Poiché questa funzionalità sincronizza i cluster database secondari con quello primario prima di apportare altre modifiche, l'RPO è effettivamente pari a 0 (nessuna perdita di dati). Per informazioni, consultare Esecuzione di failover pianificati gestiti per database globali Neptune.

Scollegamento e promozione di un database globale Neptune in caso di interruzione non pianificata

Nella raro caso in cui il database globale Neptune subisca un'interruzione imprevista nella regione Regione AWS primaria, il cluster database Neptune primario e il relativo nodo di scrittura

diventano non disponibili e la replica tra il cluster primario e i cluster secondari cessa. Per ridurre al minimo i tempi di inattività (RTO) e la perdita di dati (RPO), è possibile eseguire un'operazione di scollegamento e promozione tra regioni per ricostruire il database globale.

> **Tip**
>
> È consigliabile acquisire familiarità con questo processo prima di utilizzarlo e predisporre un piano per procedere rapidamente al primo segnale di un problema a livello regionale.
>
> - Usa Amazon CloudWatch regolarmente per monitorare i tempi di ritardo dei cluster secondari in modo identificare la regione secondaria con il minor tempo di ritardo per un eventuale failover.
> - Assicurati di testare il piano per verificare che le procedure siano complete e accurate.
> - Utilizza un ambiente simulato per assicurarti che il personale sia formato e pronto a eseguire rapidamente un failover di ripristino di emergenza, se necessario.

Per eseguire il failover su un cluster secondario dopo un'interruzione non pianificata nell'area principale

1. Arresta l'emissione di query di mutazione e di altre operazioni di scrittura sul cluster database primario.
2. Identifica un cluster database in una Regione AWS secondaria da usare come nuovo cluster database primario del database globale. Se nel database globale sono presenti due o più Regioni AWS secondarie, scegli il cluster secondario con il minor tempo di ritardo.
3. Scollega il cluster database secondario scelto dal database globale Neptune.

   La rimozione di un cluster database secondario da un database globale Neptune interrompe immediatamente la replica dal cluster primario a quello secondario e lo promuove a un cluster database autonomo con funzionalità di lettura/scrittura complete. Gli altri cluster secondari nel database globale saranno ancora disponibili e potranno accettare chiamate di lettura dall'applicazione.

   Prima di ricreare il database globale Neptune, dovrai anche scollegare gli altri cluster secondari per evitare incoerenze dei dati dati tra i cluster (consulta Rimozione di un cluster).
4. Riconfigura l'applicazione per inviare tutte le operazioni di scrittura al cluster database Neptune autonomo che hai scelto come nuovo cluster primario, utilizzando il nuovo endpoint

corrispondente. Se sono stati accettati i nomi predefiniti al momento della creazione del database globale Neptune, puoi modificare l'endpoint rimuovendo `-ro` dalla stringa endpoint del cluster nell'applicazione.

Ad esempio, l'endpoint del cluster secondario `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com` diventa `my-global.cluster-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com` quando tale cluster viene scollegato dal database globale.

Questo cluster database diventa il cluster primario di un nuovo database globale Neptune quando si inizia ad aggiungervi le regioni nel passaggio successivo.

5. Aggiungi una Regione AWS al cluster di database. Quando esegui questa operazione, inizia il processo di replica da primario a secondario. Per informazioni, consultare Aggiunta di regioni di database globale secondarie a una regione primaria in Amazon Neptune .
6. Aggiungi altre Regioni AWS in base alle esigenze per ricreare la topologia necessaria per supportare l'applicazione.

Assicurati che le scritture delle applicazioni vengano inviate al cluster database Neptune corretto prima, durante e dopo aver apportato queste modifiche. In questo modo, si evitano incoerenze dei dati tra i cluster database nel database globale Neptune (errori split-brain).

Esecuzione di failover pianificati gestiti per database globali Neptune

Il failover pianificato gestito consente di trasferire il cluster primario del database globale Neptune in un'altra Regione AWS in base alle esigenze. Alcune organizzazioni decidono di ruotare regolarmente le regioni dei cluster primari.

> **Note**
>
> Il processo di failover pianificato gestito descritto qui è progettato per essere utilizzato in un database globale Neptune integro. Per eseguire un ripristino da un'interruzione non pianificata o per eseguire un test di ripristino di emergenza, segui il processo di scollegamento e promozione.

Durante un failover pianificato gestito, per il cluster primario viene eseguito il failover nella regione secondaria scelta, mentre la topologia di replica esistente del database globale viene mantenuta. Prima dell'inizio del processo di failover pianificato gestito, il database globale sincronizza tutti i cluster secondari con il relativo cluster primario. Dopo aver verificato che tutti i cluster siano

sincronizzati, il failover pianificato gestito può iniziare. Il cluster database nella regione primaria diventa di sola lettura e il cluster secondario scelto promuove una delle sue istanze di sola lettura allo stato di scrittura completo, consentendo così al cluster di assumere il ruolo di cluster primario. Poiché tutti i cluster secondari sono stati sincronizzati con quello primario all'inizio del processo, il nuovo cluster primario continua le operazioni per il database globale senza perdere alcun dato. Il database non è disponibile per un breve periodo, mentre i cluster primario e secondario selezionati assumono i rispettivi nuovi ruoli.

Per ottimizzare la disponibilità delle applicazioni, puoi eseguire il failover durante le ore non di punta, in un momento in cui le scritture nel cluster database primario sono minime. Inoltre, procedi come indicato di seguito prima di avviare il failover:

- Porta le applicazioni offline laddove possibile per ridurre le scritture sul cluster primario.
- Controlla i tempi di ritardo per tutti i cluster database Neptune secondari nel database globale e scegli il cluster secondario con il minor ritardo complessivo per la promozione a primario. Utilizza Amazon CloudWatch per visualizzare il parametro `NeptuneGlobalDBProgressLag` per tutti i secondari. Questa metrica indica quanto è indietro (in millisecondi) un cluster secondario rispetto al cluster database primario. Il suo valore è direttamente proporzionale al tempo necessario a Neptune per completare il failover. In altre parole, maggiore è il valore di ritardo, più lunga è l'interruzione dovuta al failover, quindi è consigliabile scegliere la regione con il ritardo minore. Per ulteriori informazioni, consulta Metriche di Neptune CloudWatch .

Durante un failover pianificato gestito, il cluster database secondario scelto viene promosso al nuovo ruolo di cluster primario ma non eredita la configurazione completa del cluster database primario. Una mancata corrispondenza nella configurazione può causare problemi di prestazioni, incompatibilità dei carichi di lavoro e altri comportamenti anomali. Per evitare tali problemi, risolvi le differenze di configurazione seguenti tra i cluster database globali prima del failover:

- Configura i parametri nel nuovo cluster primario affinché corrispondano a quelli del cluster primario corrente.
- Configura gli strumenti di monitoraggio, le opzioni e gli allarmi: configura il cluster database che diventerà il nuovo cluster primario con le stesse funzionalità di registrazione, gli stessi allarmi e le stesse opzioni attualmente impostati nel cluster primario corrente.
- Configura le integrazioni con altri servizi AWS: se il database globale Neptune si integra con i servizi AWS, ad esempio AWS Identity and Access Management (IAM), Amazon S3 o AWS Lambda, è necessario assicurarsi che questi siano configurati in modo appropriato per l'integrazione con il nuovo cluster database primario.

Quando il processo di failover è completato e il cluster database promosso è pronto per gestire le operazioni di scrittura per il database globale, assicurati di modificare le applicazioni affinché usino il nuovo endpoint per il nuovo cluster primario.

Utilizzo della AWS CLI per avviare un failover pianificato gestito

Usa il comando CLI failover-global-cluster (che racchiude l'API FailoverGlobalCluster) per eseguire il failover del database globale Neptune:

```
aws neptune failover-global-cluster \
  --region (the region where the primary cluster is located) \
  --global-cluster-identifier (global database ID) \
  --target-db-cluster-identifier (the ARN of the secondary DB cluster to promote)
```

> **Note**
>
> L'API `failover-global-cluster` non è disponibile nell'anteprima. Farà parte del rilascio disponibile a livello generale.

Monitoraggio di un database globale Neptune con le metriche di CloudWatch

Neptune supporta le seguenti metriche di CloudWatch che puoi utilizzare per monitorare un database globale Neptune:

- **`GlobalDbDataTransferBytes`**: numero di byte di dati del log di ripristino trasferiti dalla Regione AWS primaria a una Regione AWS secondaria in un database globale Neptune.
- **`GlobalDbReplicatedWriteIO`**: numero di operazioni di I/O di scrittura replicate dalla regione Regione AWS primaria nel database globale al volume cluster in una Regione AWS secondaria.

  I calcoli di fatturazione per ogni cluster database in un database globale Neptune utilizzano la metrica `VolumeWriteIOPS` per determinare il numero di scritture eseguite all'interno del cluster. Per il cluster database primario, i calcoli di fatturazione usano `NeptuneGlobalDbReplicatedWriteIO` per tenere conto della replica tra regioni nei cluster database secondari.
- **`GlobalDbProgressLag`**: numero di millisecondi di ritardo del cluster secondario rispetto al cluster primario sia per le transazioni utente che per le transazioni di sistema.

Parametro	Dimensione	Pubblicato in	Unità
`GlobalDbDataTransferBytes`	SourceRegion, DBClusterIdentifier	Secondary	Byte
`GlobalDbReplicatedWriteIO`	SourceRegion, DBClusterIdentifier	Secondary	Conteggio
`GlobalDbProgressLag`	DBClusterIdentifier, SecondaryRegion : in Primary DBCluster Identifier, SourceReg ion : in Secondary	Primaria, secondaria	Millisecondi

Panoramica delle funzionalità di Amazon Neptune

In questa sezione viene fornita una panoramica delle funzionalità specifiche di Neptune, tra cui:

- Conformità di Neptune agli standard dei linguaggi di query.
- Modello di dati a grafo di Neptune.
- Spiegazione della semantica delle transazioni di Neptune.
- Introduzione ai cluster e alle istanze di Neptune.
- Archiviazione, affidabilità e disponibilità di Neptune.
- Spiegazione degli endpoint di Neptune.
- Come gli ID query personalizzati di Neptune consentono di controllare lo stato delle query.
- Utilizzo della modalità di laboratorio di Neptune per abilitare funzionalità sperimentali.
- Descrizione del motore DFE di Neptune.
- Connettività JDBC di Neptune.
- Elenco dei rilasci del motore Neptune e come aggiornare il motore.

> **Note**
>
> Questa sezione non tratta l'uso dei linguaggi di query che è possibile utilizzare per accedere ai dati in un grafo Neptune.
> Per informazioni su come connettersi a un cluster database Neptune in esecuzione con Gremlin, consulta Accesso al grafo Neptune con Gremlin.
> Per informazioni su come connettersi a un cluster database Neptune in esecuzione con Gremlin, consulta Accesso al grafo di Neptune con openCypher.
> Per informazioni su come connettersi a un cluster database Neptune in esecuzione con SPARQL, consulta Accesso al grafo Neptune con SPARQL.

Argomenti

- Cluster e istanze database di Amazon Neptune
- Archiviazione, affidabilità e disponibilità di Amazon Neptune
- Connessione agli endpoint Amazon Neptune
- Inserimento di un ID personalizzato in una query Neptune Gremlin o SPARQL
- Modalità di laboratorio Neptune
- Il motore di query alternativo (DFE) di Amazon Neptune
- Gestione delle statistiche utilizzabili dal motore DFE Neptune
- Creazione di un breve report di riepilogo sul grafo
- Connettività JDBC di Amazon Neptune
- Aggiornamenti del motore Amazon Neptune

Note sulla conformità agli standard di Amazon Neptune

Amazon Neptune è conforme agli standard applicabili nell'implementazione dei linguaggi di query a grafo Gremlin e SPARQL nella maggior parte dei casi.

Queste sezioni descrivono gli standard e le aree in cui Neptune si estende o si discosta da essi.

Argomenti

Conformità agli standard Gremlin in Amazon Neptune

Le seguenti sezioni forniscono una panoramica dell'implementazione di Neptune di Gremlin e di come si differenzia dall'implementazione di Apache. TinkerPop

Neptune implementa alcuni passaggi di Gremlin in modo nativo nel suo motore e utilizza l'implementazione di TinkerPop Apache Gremlin per elaborarne altri (vedi). Supporto nativo dei passaggi Gremlin in Amazon Neptune

> **Note**
>
> Per alcuni esempi concreti di queste differenze di implementazione mostrate nella console Gremlin e in Amazon Neptune, consulta la sezione the section called “Utilizzo di Gremlin” del Quick Start.

Argomenti

- Transazioni
- ID dei vertici e degli archi
- ID forniti dagli utenti
- ID di proprietà dei vertici
- Cardinalità delle proprietà dei vertici
- Aggiornamento della proprietà di un vertice
- Etichette
- Caratteri escape
- Limitazioni Groovy
- Serializzazione
- Passaggi Lambda
- Metodi Gremlin non supportati
- Passaggi Gremlin non supportati
- Caratteristiche del grafo Gremlin in Neptune

Standard applicabili per Gremlin

- Il linguaggio Gremlin è definito da Apache TinkerPop Documentation e dall'implementazione Apache di Gremlin piuttosto che da una specifica formale. TinkerPop
- Per i formati numerici, Gremlin segue lo standard IEEE 754 (IEEE 754-2019 - IEEE Standard for Floating-Point Arithmetic. Per ulteriori informazioni, consulta anche la pagina IEEE 754 di Wikipedia).

Variabili e parametri negli script

Per quanto riguarda le variabili pre-bound, l'oggetto di attraversamento g è pre-bound in Neptune e l'oggetto `graph` non è supportato.

Sebbene Neptune non supporti le variabili Gremlin o la parametrizzazione negli script, spesso si possono trovare in Internet script di esempio per Gremlin Server che contengono dichiarazioni di variabili, come ad esempio:

```
String query = "x = 1; g.V(x)";
```

```
List<Result> results = client.submit(query).all().get();
```

Esistono anche molti esempi che utilizzano la parametrizzazione (o le associazioni) quando si inviano le query, come ad esempio:

```
Map<String,Object> params = new HashMap<>();
params.put("x",1);
String query = "g.V(x)";
List<Result> results = client.submit(query).all().get();
```

Gli esempi di parametri sono in genere associati ad avvisi relativi alle penalizzazioni in termini di prestazioni in caso di mancata parametrizzazione quando possibile. Di esempi simili ce ne sono moltissimi, e tutti sembrano abbastanza convincenti sulla necessità di parametrizzare. TinkerPop

Tuttavia, sia la funzionalità di dichiarazione delle variabili che la funzionalità di parametrizzazione (insieme agli avvisi) si applicano al Gremlin Server solo quando TinkerPop utilizza il. `GremlinGroovyScriptEngine` Non si applicano quando Gremlin Server utilizza la grammatica ANTLR `gremlin-language` di Gremlin per analizzare le query. La grammatica ANTLR non supporta né le dichiarazioni di variabili né la parametrizzazione, quindi quando si utilizza ANTLR, non ci si deve preoccupare di non riuscire a parametrizzare. Poiché la grammatica ANTLR è un componente più recente di TinkerPop, i contenuti meno recenti che potresti incontrare su Internet generalmente non riflettono questa distinzione.

Neptune utilizza la grammatica ANTLR nel motore di elaborazione delle query anziché `GremlinGroovyScriptEngine`, quindi non supporta le variabili, la parametrizzazione o la proprietà `bindings`. Di conseguenza, i problemi relativi alla mancata parametrizzazione non si applicano a Neptune. Utilizzando Neptune, è perfettamente sicuro inviare semplicemente la query così com'è, dove normalmente si parametrizza. Di conseguenza, l'esempio precedente può essere semplificato senza alcuna penalizzazione delle prestazioni come segue:

```
String query = "g.V(1)";
List<Result> results = client.submit(query).all().get();
```

TinkerPop enumerazioni

Neptune non supporta nomi di classe completi per i valori di enumerazione. Ad esempio, devi usare `single` e non `org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single` nella richiesta Groovy.

Il tipo di enumerazione è determinato dal tipo di parametro.

La tabella seguente mostra i valori di enumerazione consentiti e il relativo nome completo. TinkerPop

Valori consentiti	Classe
`id`, `Chiave`, `etichetta` , `value`	org.apache.tinkerpop.gremlin.structure.T
`T.id`, `T.key`, `T.label`, `T.value`	org.apache.tinkerpop.gremlin.structure.T
`set`, `single`	org.apache.tinkerpop.gremlin.structure. VertexProperty.Cardinalità
`asc`, `desc`, `shuffle`	org.apache.tinkerpop.gremlin.process.traversal.Order
`Order.asc` , `Order.desc` , `Order.shuffle`	org.apache.tinkerpop.gremlin.process.traversal.Order
`global`, `local`	org.apache.tinkerpop.gremlin.process.traversal.Scope
`Scope.global` , `Scope.local`	org.apache.tinkerpop.gremlin.process.traversal.Scope
`tutto`, `first`, `last`, `mixed`	org.apache.tinkerpop.gremlin.process.traversal.Pop
`normSack`	org.apache.tinkerpop.gremlin.process.traversal. SackFunctions.Barriera
`addAll`, `e`, `assign`, `div`, `max`, `min`, `minus`, `mult`, `oppure`, `sum`, `sumLong`	org.apache.tinkerpop.gremlin.process.traversal.Operator
`keys`, `values`	org.apache.tinkerpop.gremlin.structure.Column
`BOTH`, `IN`, `OUT`	org.apache.tinkerpop.gremlin.structure.Direction
`any`, `nessuno`	org.apache.tinkerpop.gremlin.process.traversal.step. TraversalOptionParent.Scegli

Codice Java

Neptune non supporta le chiamate ai metodi definite da chiamate arbitrarie Java o libreria Java diverse da quelle API Gremlin supportate. Ad esempio, `java.lang.*`, `Date()`e `g.V().tryNext().orElseGet()` non sono consentite.

Esecuzione di uno script

Tutte le query devono iniziare con g, l'oggetto di attraversamento.

In Invii query stringa, possono essere emessi più attraversamenti separati da un punto e virgola (`;`) o un carattere nuova riga (\n). Per essere eseguita, ogni istruzione diversa dall'ultima deve terminare con una fase `.iterate()`. Vengono restituiti solo i dati di attraversamento finali. Nota che ciò non si applica agli invii di ByteCode query GLV.

Sessioni

Le sessioni in Neptune hanno una durata limitata di soli 10 minuti. Per ulteriori informazioni, consulta Sessioni basate su script Gremlin il riferimento alla TinkerPop sessione.

Transazioni

Neptune apre una nuova transazione all'inizio di ogni attraversamento Gremlin e chiude la transazione dopo il completamento dell'attraversamento. La transazione viene ripristinata quando si verifica un errore.

Una singola transazione include più istruzioni separate da un punto e virgola (`;`) o un carattere nuova riga (\n). Ogni istruzione diversa dall'ultima deve terminare con una fase `next()` da eseguire. Vengono restituiti solo i dati di attraversamento finali.

La logica di transazione manuale che utilizza `tx.commit()` e `tx.rollback()` non è supportata.

> **Important**
>
> Si applica solo ai metodi dove invii la query Gremlin come stringa di testo (vedi Transazioni Gremlin).

ID dei vertici e degli archi

Gli ID dei vertici e degli archi in Neptune Gremlin devono essere di tipo `String`. Queste stringhe di ID supportano i caratteri Unicode e non possono superare i 55 MB di dimensione.

Gli ID forniti dagli utenti sono supportati, ma sono facoltativi nel normale utilizzo. Se non si fornisce un ID quando si aggiunge un vertice o un arco, Neptune genera un UUID e lo converte in una stringa, in un formato simile al seguente: `"48af8178-50ce-971a-fc41-8c9a954cea62"`. Questi UUID non sono conformi allo standard RFC, quindi se sono necessari UUID standard occorre generarli esternamente e fornirli quando si aggiungono vertici o archi.

> **Note**
>
> Il comando `Load` di Neptune richiede che tutti gli ID siano specificati utilizzando il campo ~id nel formato CSV di Neptune.

ID forniti dagli utenti

Gli ID forniti dagli utenti sono consentiti in Neptune Gremlin con le condizioni seguenti.

- Gli ID forniti sono facoltativi.
- Sono supportati solo i vertici e gli edge.
- È supportato solo il tipo `String`.

Per creare un nuovo vertice con un ID personalizzato, utilizza la fase `property` con la parola chiave `id`: `g.addV().property(id, 'customid')`.

> **Note**
>
> Non mettere tra virgolette la parola chiave `id`. Si riferisce a `T.id`.

Tutti gli ID dei vertici e degli edge devono essere univoci. Tuttavia, Neptune consente che un vertice e un arco abbiano lo stesso ID.

Se si tenta di creare un nuovo vertice utilizzando `g.addV()` e un vertice con tale ID esiste già, l'operazione non va a buon fine. Fa eccezione il caso in cui si specifica una nuova etichetta per il

vertice; in tal caso l'operazione viene eseguita correttamente, ma aggiunge al vertice esistente la nuova etichetta e le eventuali proprietà aggiuntive specificate. Nessun elemento viene sovrascritto. Non viene creato un nuovo vertice. L'ID del vertice non cambia e rimane univoco.

Ad esempio, i comandi della console Gremlin riportato di seguito, hanno esito positivo:

```
gremlin> g.addV('label1').property(id, 'customid')
gremlin> g.addV('label2').property(id, 'customid')
gremlin> g.V('customid').label()
==>label1::label2
```

ID di proprietà dei vertici

Gli ID di proprietà dei vertici sono generati automaticamente e possono comparire come numeri positivi o negativi quando richiesto.

Cardinalità delle proprietà dei vertici

Neptune supporta la cardinalità di un insieme e la cardinalità singola. Se non è specificato, è impostata la cardinalità di un insieme. In questo modo, se imposti un valore di proprietà, questo aggiunge un nuovo valore alla proprietà, ma solo se non compare già nell'insieme dei valori. Questo è il valore di enumerazione Gremlin di Set.

`List` non è supportato. Per ulteriori informazioni sulla cardinalità delle proprietà, consultate l'argomento Vertex nel Gremlin. JavaDoc

Aggiornamento della proprietà di un vertice

Per aggiornare un valore di proprietà senza aggiungere un ulteriore valore all'insieme dei valori, specificare la cardinalità `single` nella fase `property`.

```
g.V('exampleid01').property(single, 'age', 25)
```

Questo consente di rimuovere tutti i valori esistenti della proprietà.

Etichette

Neptune supporta più etichette per un vertice. Quando crei un'etichetta, puoi specificare più etichette separandole con `::`. Ad esempio, `g.addV("Label1::Label2::Label3")` aggiunge un vertice con tre etichette diverse. La fase `hasLabel` corrisponde a questo vertice con una qualsiasi delle tre etichette: `hasLabel("Label1")` , `hasLabel("Label2")` e `hasLabel("Label3")`.

> **Important**
>
> Il delimitatore `::` è riservato solo a quest'uso. Non è possibile specificare più etichette nella fase `hasLabel`. Ad esempio, `hasLabel("Label1::Label2")` non corrisponde a nulla.

Caratteri escape

Neptune risolve tutti i caratteri escape come descritto nella sezione Escape dei caratteri speciali della documentazione Apache Groovy Language.

Limitazioni Groovy

Neptune non supporta i comandi Groovy che non iniziano con `g`. Questo include la matematica (ad esempio: `1+1`), le chiamate di sistema (ad esempio: `System.nanoTime()`) e le definizioni delle variabili (ad esempio: `1+1`).

> **Important**
>
> Neptune non supporta nomi di classe completi. Ad esempio, devi usare `single` e non `org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single` nella richiesta Groovy.

Serializzazione

Neptune supporta le serializzazioni seguenti in base al tipo MIME richiesto.

MIME type	Serialization	Configuration
`application/vnd.gremlin-v1.0+gryo`	`GryoMessageSerializerV1d0`	`Registri IO: [org.apache.tinkerpop.gremlin.tinkergraph.structure. TinkerIoRegistryV1d0]`
`application/vnd.gremlin-v1.0+gryo-stringd`	`GryoMessageSerializerV1d0`	`serializeResultToStringa: true}`

application/vnd.gremlin-v3.0+gryo	GryoMessageSerializerV3d0	Registri IO: [org.apache.tinkerpop.gremlin.tinkergraph.structure. TinkerIoRegistryV3d0]
application/vnd.gremlin-v3.0+gryo-stringd	GryoMessageSerializerV3d0	serializeResultToStringa: true
application/vnd.gremlin-v1.0+json	MessageSerializerGremlinGraphSon V1d0	Registri IO: [org.apache.tinkerpop.gremlin.tinkergraph.structure. TinkerIoRegistryV1d0]
application/vnd.gremlin-v2.0+json	MessageSerializerGraphSon V2d0 (only works with WebSockets)	Registri IO: [org.apache.tinkerpop.gremlin.tinkergraph.structure. TinkerIoRegistryV2d0]
application/vnd.gremlin-v3.0+json	GraphSONMessageSerializerV3d0	
application/json	MessageSerializerGraphSon V3d0	Registri IO: [org.apache.tinkerpop.gremlin.tinkergraph.structure. TinkerIoRegistryV3d0]
application/vnd.graphbinary-v1.0	GraphBinaryMessageSerializerV1	

Passaggi Lambda

Neptune non supporta i passaggi Lambda.

Metodi Gremlin non supportati

Neptune non supporta i metodi Gremlin seguenti:

- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.progra`
- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.sideEf`
- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.from(o`
- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.to(org`

Ad esempio, l'attraversamento seguente non è consentito: `g.V().addE('something').from(__.V().next()).to(__.V().next())`.

> **Important**
>
> Si applica solo ai metodi dove invii la query Gremlin come stringa di testo.

Passaggi Gremlin non supportati

Neptune non supporta i passaggi Gremlin seguenti:

- Il passaggio io( ) Gremlin è supportata solo parzialmente in Neptune. Può essere usata in un contesto di lettura, come in `g.io(`*`url`*`).read()`, ma non in un contesto di scrittura.

Caratteristiche del grafo Gremlin in Neptune

L'implementazione di Gremlin in Neptune non espone l'oggetto `graph`. Le tabelle seguenti elencano le caratteristiche di Gremlin e indicano se Neptune le supporta o meno.

Supporto di Neptune delle funzionalità di **`graph`**

Le caratteristiche del grafo Neptune, se supportate, sono le stesse di quelle restituite dal comando `graph.features()`.

Caratteristica del grafo	Abilitata?
`Transazioni`	true

`ThreadedTransactions`	false
`Computer`	false
`Persistenza`	true
`ConcurrentAccess`	true

Supporto di Neptune delle funzionalità della variabile

Caratteristica della variabile	Abilitata?
`Variables`	false
`SerializableValues`	false
`UniformListValues`	false
`BooleanArrayValues`	false
`DoubleArrayValues`	false
`IntegerArrayValues`	false
`StringArrayValues`	false
`BooleanValues`	false
`ByteValues`	false
`DoubleValues`	false
`FloatValues`	false
`IntegerValues`	false
`LongValues`	false
`MapValues`	false
`MixedListValues`	false

`StringValues`	false
`ByteArrayValues`	false
`FloatArrayValues`	false
`LongArrayValues`	false

Supporto di Neptune delle caratteristiche del vertice

Caratteristica del vertice	Abilitata?
`MetaProperties`	false
`DuplicateMultiProperties`	false
`AddVertices`	true
`RemoveVertices`	true
`MultiProperties`	true
`UserSuppliedIds`	true
`AddProperty`	true
`RemoveProperty`	true
`NumericIds`	false
`StringIds`	true
`UuidIds`	false
`CustomIds`	false
`AnyIds`	false

Supporto di Neptune delle caratteristiche della proprietà del vertice

Caratteristica della proprietà del vertice	Abilitata?
`UserSuppliedIds`	false
`AddProperty`	true
`RemoveProperty`	true
`NumericIds`	true
`StringIds`	true
`UuidIds`	false
`CustomIds`	false
`AnyIds`	false
`Proprietà`	true
`SerializableValues`	false
UniformListValues	false
`BooleanArrayValues`	false
`DoubleArrayValues`	false
`IntegerArrayValues`	false
`StringArrayValues`	false
`BooleanValues`	true
`ByteValues`	true
`DoubleValues`	true
`FloatValues`	true
`IntegerValues`	true

`LongValues`	true
`MapValues`	false
`MixedListValues`	false
`StringValues`	true
`ByteArrayValues`	false
`FloatArrayValues`	false
`LongArrayValues`	false

Supporto di Neptune delle caratteristiche dell'arco

Caratteristica dell'arco	Abilitata?
`AddEdges`	true
`RemoveEdges`	true
`UserSuppliedIds`	true
`AddProperty`	true
`RemoveProperty`	true
`NumericIds`	false
`StringIds`	true
`UuidIds`	false
`CustomIds`	false
`AnyIds`	false

Supporto di Neptune delle caratteristiche della proprietà dell'arco

Caratteristica della proprietà dell'arco	Abilitata?
`Proprietà`	true
`SerializableValues`	false
`UniformListValues`	false
`BooleanArrayValues`	false
`DoubleArrayValues`	false
`IntegerArrayValues`	false
`StringArrayValues`	false
`BooleanValues`	true
`ByteValues`	true
`DoubleValues`	true
`FloatValues`	true
`IntegerValues`	true
`LongValues`	true
`MapValues`	false
`MixedListValues`	false
`StringValues`	true
`ByteArrayValues`	false
`FloatArrayValues`	false
`LongArrayValues`	false

Conformità agli standard SPARQL in Amazon Neptune

Dopo aver elencato gli standard SPARQL applicabili, le seguenti sezioni forniscono dettagli specifici su come l'implementazione SPARQL di Neptune si estende o si discosta da tali standard.

Argomenti

Amazon Neptune è conforme ai seguenti standard nell'implementazione del linguaggio di query a grafo SPARQL.

Standard applicabili per SPARQL

- SPARQL è definito dalla raccomandazione W3C SPARQL 1.1 Query Language del 21 marzo 2013.
- Il protocollo SPARQL Update e il linguaggio di query sono definiti dalla specifica W3C di SPARQL 1.1 Update.
- Per i formati numerici, SPARQL segue la specifica W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes conforme alla specifica IEEE 754 (IEEE 754-2019 - IEEE Standard for Floating-Point Arithmetic. Per ulteriori informazioni, vedi anche la pagina di Wikipedia IEEE 754). Tuttavia, le funzionalità introdotte dopo la versione `IEEE 754-1985` non sono incluse nella specifica.

Prefissi dello spazio dei nomi predefiniti in Neptune SPARQL

Neptune definisce i seguenti prefissi per impostazione predefinita per l'utilizzo nelle query SPARQL. Per ulteriori informazioni, vedere Prefixed Names (Nomi con prefisso) nella specifica SPARQL.

- `rdf` – `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
- `rdfs` – `http://www.w3.org/2000/01/rdf-schema#`
- `owl` – `http://www.w3.org/2002/07/owl#`
- `xsd` – `http://www.w3.org/2001/XMLSchema#`

Grafo predefinito SPARQL e grafi denominati

Amazon Neptune associa ogni tripla a un grafo denominato. Il grafo predefinito è definito come l'unione di grafi denominati.

Grafico predefinito per query

Se invii una query SPARQL senza specificare un grafo in maniera esplicita tramite la parola chiave `GRAPH` o costrutti quali `FROM NAMED`, Neptune considera sempre tutte le triple nell'istanza database. Ad esempio, la seguente query restituisce tutte le triple da un endpoint SPARQL Neptune:

```
SELECT * WHERE { ?s ?p ?o }
```

Le triple che appaiono in più grafici vengono restituite una sola volta.

Per informazioni sulla specifica del grafico predefinito, consulta la sezione RDF Dataset delle specifiche SPARQL 1.1 Query Language.

Specifica del grafo denominato per caricamento, inserimenti o aggiornamenti

Se non specifichi un grafo denominato durante il caricamento, l'inserimento o l'aggiornamento di triple, Neptune utilizza il grafo denominato di fallback definito dall'URI `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Quando invii una richiesta `Load` Neptune utilizzando un formato basato sulla tripla, puoi specificare il grafo nominato da utilizzare per tutte le triple utilizzando il parametro `parserConfiguration: namedGraphUri`. Per ulteriori informazioni sulla sintassi del comando `Load`, consulta the section called “Comando dello strumento di caricamento”.

> **Important**
>
> Se non utilizzi questo parametro e non specifichi un grafo denominato, viene utilizzato l'URI di fallback: `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Questo grafo di fallback denominato viene anche utilizzato se carichi triple tramite `SPARQL UPDATE` senza fornire in maniera esplicita una destinazione del grafo denominato.

Puoi utilizzare il formato basato su quadruple N-Quads per specificare un grafo denominato per ogni tripla nel database.

> **Note**
>
> Utilizzare N-Quads consente di lasciare il grafo denominato vuoto. In questo caso, viene utilizzato `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`. Puoi sostituire il grafo denominato predefinito per N-Quads utilizzando l'opzione di configurazione parser `namedGraphUri`.

Funzioni costruttore SPARQL XPath supportate da Neptune

Lo standard SPARQL consente ai motori SPARQL di supportare un insieme estensibile di funzioni costruttore XPath. Neptune attualmente supporta le seguenti funzioni costruttore, dove il prefisso `xsd` è definito come `http://www.w3.org/2001/XMLSchema#`:

- `xsd:boolean`
- `xsd:integer`
- `xsd:double`
- `xsd:float`
- `xsd:decimal`
- `xsd:long`
- `xsd:unsignedLong`

IRI di base predefinito per query e aggiornamenti

Poiché un cluster Neptune ha diversi endpoint, l'utilizzo dell'URL di richiesta di una query o di un aggiornamento come IRI di base potrebbe portare a risultati imprevisti durante la risoluzione degli IRI relativi.

A partire dal rilascio 1.2.1.0 del motore, Neptune utilizza `http://aws.amazon.com/neptune/default/` come IRI di base se un IRI di base esplicito non fa parte della richiesta.

Nella richiesta seguente, l'IRI di base fa parte della richiesta:

```
BASE <http://example.org/default/>
INSERT DATA { <node1> <id> "n1" }

BASE <http://example.org/default/>
SELECT * { <node1> ?p ?o }
```

E il risultato sarà:

```
?p                                                  ?o
http://example.org/default/id                       n1
```

In questa richiesta, tuttavia, non è incluso alcun IRI di base:

```
INSERT DATA { <node1> <id> "n1" }

SELECT * { <node1> ?p ?o }
```

In tal caso, il risultato sarà:

```
?p                                                  ?o
http://aws.amazon.com/neptune/default/id            n1
```

Valori xsd:dateTime in Neptune

Per motivi di prestazioni, Neptune archivia sempre i valori di data/ora come Tempo coordinato universale (UTC). Questo rende i confronti diretti molto efficienti.

Ciò significa anche che se si immette un valore `dateTime` che specifica un particolare fuso orario, Neptune converte il valore in formato UTC ed elimina le informazioni sul fuso orario. Quindi, quando si recupera il valore `dateTime` in un secondo momento, viene espresso in formato UTC, non l'ora del fuso orario originale e non è più possibile risalire al fuso orario originale.

Gestione di Neptune dei valori speciali a virgola mobile

Neptune gestisce i valori speciali a virgola mobile in SPARQL nel seguente modo:

Gestione del valore NaN di SPARQL in Neptune

In Neptune, SPARQL può accettare un valore `NaN` in una query. Non viene fatta alcuna distinzione tra valori `NaN` di segnalazione e silenziosi. Neptune considera tutti i valori `NaN` come silenziosi.

Dal punto di vista semantico, non è possibile un confronto di un valore `NaN` poiché non esistono elementi maggiori di, minori di o uguali a un valore `NaN`. Ciò significa che un valore `NaN` su un lato di un confronto in teoria non corrisponde mai a qualcosa sull'altro lato.

Tuttavia, la specifica XSD considera due valori `NaN` `xsd:double` o `xsd:float` come valori uguali. Neptune segue questo principio per il filtro `IN`, per l'operatore equal nelle espressioni di filtro e per la semantica di corrispondenza esatta (avendo un valore `NaN` nella posizione dell'oggetto di un modello di tripla).

Gestione del valore infinito di SPARQL in Neptune

In Neptune, SPARQL può accettare un valore `INF` o `-INF` in una query. `INF` viene confrontato come maggiore di qualsiasi altro valore numerico e `-INF` viene confrontato come minore di qualsiasi altro valore numerico.

Due valori INF con segni corrispondenti si confrontano come uguali tra loro indipendentemente dal loro tipo (ad esempio, un float `-INF` confronta come uguale a un doppio `-INF`).

Non è possibile un confronto con un valore `NaN` ovviamente poiché non esistono elementi maggiori di, minori di o uguali a un valore `NaN`.

Gestione dello zero negativo di SPARQL in Neptune

Neptune normalizza un valore zero negativo a uno zero senza segno. È possibile utilizzare i valori zero negativo in una query, ma non vengono registrati come tali nel database e vengono confrontati come uguali a zero senza segno.

Limitazione di Neptune dei valori di lunghezza arbitraria

Neptune limita la dimensione di archiviazione dei valori interi XSD, dei valori a virgola mobile e dei valori decimali in SPARQL a 64 bit. L'utilizzo di valori più grandi genera un errore `InvalidNumericDataException`.

Neptune estende il confronto di uguaglianza in SPARQL

Lo standard SPARQL definisce una logica ternaria per le espressioni di valore, in cui un'espressione di valore può restituire `true`, `false`, o `error`. La semantica predefinita per l'uguaglianza dei termini definita nella specifica SPARQL 1.1, che si applica ai confronti = e `!=` nelle condizioni `FILTER`, produce un `error` quando si confrontano tipi di dati che non sono esplicitamente comparabili nella tabella degli operatori nella specifica.

Questo comportamento può portare a risultati non intuitivi, come nell'esempio seguente.

Dati:

```
<http://example.com/Server/1> <http://example.com/ip> "127.0.0.1"^^<http://example.com/
datatype/IPAddress>
```

Query 1:

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o = "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

Query 2:

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o != "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

Con la semantica SPARQL predefinita utilizzata da Neptune prima del rilascio 1.0.2.1, entrambe le query restituiranno il risultato vuoto. Il motivo è che `?o = "127.0.0.2"^^<http://example.com/IPAddress>` quando viene valutato per `?o := "127.0.0.1"^^<http://example.com/IPAddress>`, produce un `error` anziché `false` perché non esistono regole di confronto esplicite specificate per il tipo di dati personalizzato `<http://example.com/IPAddress>`. Di conseguenza, anche la versione negata nella seconda query produce un `error`. In entrambe le query, l'`error` causa il filtraggio della soluzione candidata.

A partire dal rilascio 1.0.2.1, Neptune ha esteso l'operatore di disuguaglianza SPARQL in accordo con le specifiche. Vedere la sezione SPARQL 1.1 sull'estensibilità dell'operatore, che consente ai motori di definire regole aggiuntive su come eseguire il confronto tra tipi di dati incorporati definiti dall'utente e non comparabili.

Utilizzando questa opzione, Neptune ora tratta un confronto di due tipi di dati che non è esplicitamente definito nella tabella di mappatura dell'operatore come se restituisse `true`, se i valori letterali e i tipi di dati sono sintatticamente uguali, altrimenti, false. In ogni caso, non viene prodotto un `error`.

Usando queste nuove semantiche, la seconda query restituirebbe `"127.0.0.1"^^<http://example.com/IPAddress>` invece di un risultato vuoto.

Gestione di valori letterali Out-of-Range (Fuori intervallo) in Neptune SPARQL

La semantica XSD definisce ogni tipo numerico con il suo spazio valore, ad eccezione di `integer` e `decimal`. Queste definizioni limitano ogni tipo a un intervallo di valori. Ad esempio, l'intervallo di un intervallo `xsd:byte` è compreso tra -128 e +127, inclusi. Qualsiasi valore al di fuori di questo intervallo è considerato non valido.

Se si tenta di assegnare un valore letterale al di fuori dello spazio dei valori di un tipo (ad esempio, se si tenta di impostare an su un `xsd:byte` valore letterale di 999), Neptune accetta il valore così com'è, senza arrotondarlo o out-of-range troncarlo. Ma non lo mantiene come valore numerico perché il tipo fornito non può rappresentarlo.

Cioè, Neptune accetta `"999"^^xsd:byte` anche se è un valore al di fuori dell'intervallo di valori `xsd:byte` definito. Tuttavia, dopo che il valore viene mantenuto nel database, può essere utilizzato solo nella semantica di corrispondenza esatta, in una posizione oggetto di un modello triplo. Nessun filtro di intervallo può essere eseguito su di esso perché i valori letterali non vengono trattati come valori numerici. out-of-range

La specifica SPARQL 1.1 definisce gli operatori di intervallo nel formato `numeric`-operator-`numeric`, `string`-operator-`string`, `literal`-operator-`literal` e così via. Neptune non può eseguire un operatore di confronto di intervalli simile a `invalid-literal`-operator-`numeric-value`.

Conformità alle specifiche OpenCypher in Amazon Neptune

Il rilascio di Amazon Neptune di openCypher supporta generalmente le clausole, gli operatori, le espressioni, le funzioni e la sintassi definiti nella specifica openCypher corrente, ovvero Cypher Query Language Reference versione 9. Le limitazioni e le differenze nel supporto di Neptune per openCypher sono indicate di seguito.

> **Note**
>
> L'attuale implementazione Neo4j di Cypher contiene funzionalità che non sono contenute nella specifica openCypher menzionata sopra. Se stai migrando il codice Cypher corrente a Neptune, consulta Compatibilità di Neptune con Neo4j e Riscrittura delle query Cypher da eseguire in openCypher su Neptune per ulteriori informazioni.

Supporto delle clausole openCypher in Neptune

Neptune supporta le seguenti clausole, ad eccezione di quanto indicato:

- `MATCH`: supportata, ad eccezione di *`shortestPath()`* e *`allShortestPaths()`* che non sono attualmente supportate.
- `OPTIONAL MATCH`
- *`MANDATORY MATCH`*: non è attualmente supportata in Neptune. Neptune, tuttavia, supporta valori ID personalizzati nelle query `MATCH`.
- `RETURN`: supportata, tranne quando viene utilizzata con valori non statici per `SKIP`o `LIMIT`. Ad esempio, quanto segue attualmente non funziona:

```
MATCH (n)
RETURN n LIMIT toInteger(rand())    // Does NOT work!
```

- `WITH`: supportata, tranne quando viene utilizzata con valori non statici per `SKIP`o `LIMIT`. Ad esempio, quanto segue attualmente non funziona:

```
MATCH (n)
WITH n SKIP toInteger(rand())
WITH count() AS count
RETURN count > 0 AS nonEmpty    // Does NOT work!
```

- `UNWIND`
- `WHERE`
- `ORDER BY`
- `SKIP`
- `LIMIT`
- `CREATE`: Neptune consente di creare valori ID personalizzati nelle query `CREATE`.
- `DELETE`
- `SET`
- `REMOVE`
- `MERGE`: Neptune supporta valori ID personalizzati nelle query `MERGE`.
- *`CALL[YIELD...]`*: non è attualmente supportata in Neptune.
- `UNION, UNION ALL`: le query di sola lettura sono supportate, ma le query di mutazione non sono attualmente supportate.

Supporto degli operatori openCypher in Neptune

Neptune supporta i seguenti operatori, ad eccezione di quanto indicato:

Operatori generali

- `DISTINCT`
- Operatore **`.`** per accedere alle proprietà di una mappa letterale annidata.

Operatori matematici

- Operatore di addizione **`+`**.
- Operatore di sottrazione **`-`**.
- Operatore di moltiplicazione **`*`**.
- Operatore di divisione **`/`**.
- Operatore di divisione modulo **`%`**.
- L'operatore esponenziale **`^`** *`NON è supportato`*.

Operatori di confronto

- Operatore di addizione **`=`**.
- Operatore di disuguaglianza **`<>`**.
- L'operatore minore di **`<`** è supportato tranne quando uno degli argomenti è un percorso, un elenco o una mappa.
- L'operatore maggiore di **`>`** è supportato tranne quando uno degli argomenti è un percorso, un elenco o una mappa.
- L'operatore **`<=`** less-than-or-equal -to è supportato tranne quando uno degli argomenti è un percorso, una lista o una mappa.
- L'operatore **`>=`** greater-than-or-equal -to è supportato tranne quando uno degli argomenti è Path, List o Map.
- `IS NULL`
- `IS NOT NULL`
- `STARTS WITH` è supportato se i dati da cercare sono una stringa.

- `ENDS WITH` è supportato se i dati da cercare sono una stringa.
- `CONTAINS` è supportato se i dati da cercare sono una stringa.

Operatori booleani

- `AND`
- `OR`
- `XOR`
- `NOT`

Operatori di stringa

- Operatore di concatenazione **+**.

Operatori di elenco

- Operatore di concatenazione **+**.
- `IN` (verifica la presenza di un elemento nell'elenco)

Supporto delle espressioni openCypher in Neptune

Neptune supporta le seguenti espressioni, ad eccezione di quanto indicato:

- `CASE`
- L'espressione **`[]`** non è attualmente supportata in Neptune per l'accesso alle chiavi di proprietà calcolate dinamicamente all'interno di un nodo, una relazione o una mappa. Ad esempio, quanto segue non funziona:

```
MATCH (n)
WITH [5, n, {key: 'value'}] AS list
RETURN list[1].name
```

Supporto delle funzioni openCypher in Neptune

Neptune supporta le seguenti funzioni, ad eccezione di quanto indicato:

Funzioni di predicato

- `exists()`

Funzioni scalari

- `coalesce()`
- `endNode()`
- `epochmillis()`
- `head()`
- `id()`
- `last()`
- `length()`
- `randomUUID()`
- `properties()`
- `removeKeyFromMap`
- `size()`: questo metodo di overload attualmente funziona solo per espressioni di modelli, elenchi e stringhe
- `startNode()`
- `timestamp()`
- `toBoolean()`
- `toFloat()`
- `toInteger()`
- `type()`

Funzioni di aggregazione

- `avg()`
- `collect()`
- `count()`
- `max()`
- `min()`

- `percentileDisc()`
- `stDev()`
- `percentileCont()`
- `stDevP()`
- `sum()`

Elencare le funzioni

- `join()` (concatena le stringhe di un elenco in un'unica stringa)
- `keys()`
- `labels()`
- `nodes()`
- `range()`
- `relationships()`
- `reverse()`
- `tail()`

Funzioni matematiche – numeriche

- `abs()`
- `ceil()`
- `floor()`
- `rand()`
- `round()`
- `sign()`

Funzioni matematiche – logaritmiche

- `e()`
- `exp()`
- `log()`
- `log10()`

- `sqrt()`

Funzioni matematiche – trigonometriche

- `acos()`
- `asin()`
- `atan()`
- `atan2()`
- `cos()`
- `cot()`
- `degrees()`
- `pi()`
- `radians()`
- `sin()`
- `tan()`

Funzioni stringa

- `join()` (concatena le stringhe di un elenco in un'unica stringa)
- `left()`
- `lTrim()`
- `replace()`
- `reverse()`
- `right()`
- `rTrim()`
- `split()`
- `substring()`
- `toLower()`
- `toString()`
- `toUpper()`
- `trim()`

Funzioni definite dall'utente

Le *funzioni definite dall'utente* non sono attualmente supportate in Neptune.

Dettagli sull'implementazione di openCypher specifici per Neptune

Le sezioni seguenti descrivono i modi in cui l'implementazione di openCypher in Neptune può differire o andare oltre la specifica openCypher.

Valutazioni di percorsi a lunghezza variabile (VLP) in Neptune

Le valutazioni dei percorsi a lunghezza variabile (VLP) rilevano i percorsi tra i nodi del grafo. La lunghezza del percorso può essere illimitata in una query. Per evitare cicli, la specifica openCypher specifica che ogni arco deve essere attraversato al massimo una volta per soluzione.

Per i VLP, l'implementazione Neptune si discosta dalla specifica openCypher in quanto supporta solo valori costanti per i filtri di uguaglianza delle proprietà. Eseguire la seguente query:

```
MATCH (x)-[:route*1..2 {dist:33, code:x.name}]->(y) return x,y
```

Poiché il valore del filtro di uguaglianza della proprietà `x.name` non è una costante, questa query restituisce l'eccezione `UnsupportedOperationException` con il messaggio: `Property predicate over variable-length relationships with non-constant expression is not supported in this release.`

Supporto temporale nell'implementazione di Neptune openCypher

Neptune attualmente fornisce un supporto limitato per la funzione temporale in openCypher. Supporta il tipo di dati `DateTime` per i tipi temporali.

La funzione `datetime()` può essere utilizzata per ottenere la data e l'ora UTC correnti in questo modo:

```
RETURN  datetime() as res
```

I valori di data e ora possono essere convertiti dai dati archiviati in Neptune in questo modo:

```
MATCH (n) RETURN datetime(n.createdDate)
```

I valori di data e ora possono essere analizzati da stringhe in formato "dateTtime" in cui data e ora sono entrambe espresse in uno dei formati supportati di seguito:

Formati di data supportati

- `yyyy-MM-dd`
- `yyyyMMdd`
- `yyyy-MM`
- `yyyy-DDD`
- `yyyyDDD`
- `yyyy`

Formati di ora supportati

- `HH:mm:ssZ`
- `HHmmssZ`
- `HH:mm:ssZ`
- `HH:mmZ`
- `HHmmZ`
- `HHZ`
- `HHmmss`
- `HH:mm:ss`
- `HH:mm`
- `HHmm`
- `HH`

Per esempio:

```
RETURN datetime('2022-01-01T00:01')      // or another example:
RETURN datetime('2022T0001')
```

Nota che tutti i valori di data/ora in Neptune openCypher vengono archiviati e recuperati come valori UTC.

Neptune openCypher utilizza un orologio `statement`, il che significa che lo stesso istante di tempo viene utilizzato per tutta la durata di una query. Una query diversa all'interno della stessa transazione può utilizzare un istante di tempo diverso.

Neptune non supporta l'uso di una funzione all'interno di una chiamata a `datetime()`. Ad esempio, quanto segue non funziona:

```
CREATE (:n {date:datetime(tostring(2021))})  // ---> NOT ALLOWED!
```

Neptune supporta la funzione `epochmillis()` che converte `datetime` in `epochmillis`. Per esempio:

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

Neptune attualmente non supporta altre funzioni e operazioni sugli oggetti `DateTime`, come l'addizione e la sottrazione.

Differenze nella semantica del linguaggio openCypher di Neptune

Neptune rappresenta gli ID dei nodi e delle relazioni come stringhe anziché come numeri interi. L'ID è uguale all'ID fornito tramite lo strumento di caricamento in blocco di dati. Se esiste uno spazio dei nomi per la colonna, lo spazio dei nomi più l'ID. Di conseguenza, la funzione `id` restituisce una stringa anziché un numero intero.

Il tipo di dati `INTEGER` è limitato a 64 bit. Quando si convertono valori a virgola mobile o stringa più grandi in un numero intero utilizzando la funzione `TOINTEGER`, i valori negativi vengono troncati a `LLONG_MIN` e i valori positivi vengono troncati a `LLONG_MAX`.

Per esempio:

```
RETURN TOINTEGER(2^100)
>  9223372036854775807

RETURN TOINTEGER(-1 * 2^100)
>  -9223372036854775808
```

Funzione **`join()`** specifica di Neptune

Neptune implementa una funzione `join()` che non è presente nella specifica openCypher. Crea un valore letterale di stringa da un elenco di valori letterali di stringa e da un delimitatore di stringa. Accetta due argomenti:

- Il primo argomento è un elenco di valori letterali di stringa.

- Il secondo argomento è la stringa del delimitatore, che può essere composta da zero, uno o più caratteri.

Esempio:

```
join(["abc", "def", "ghi"], ", ")    // Returns "abc, def, ghi"
```

Funzione **`removeKeyFromMap()`** specifica di Neptune

Neptune implementa una funzione `removeKeyFromMap()` che non è presente nella specifica openCypher. Rimuove una chiave specificata da una mappa e restituisce la nuova mappa risultante.

La funzione accetta due argomenti:

- Il primo argomento è la mappa da cui rimuovere la chiave.
- Il primo argomento è la chiave da rimuovere dalla mappa.

La funzione `removeKeyFromMap()` è particolarmente utile in situazioni in cui si desidera impostare valori per un nodo o una relazione utilizzando un elenco di mappe. Per esempio:

```
UNWIND [{`~id`: 'id1', name: 'john'}, {`~id`: 'id2', name: 'jim'}] as val
CREATE (n {`~id`: val.`~id`})
SET n = removeKeyFromMap(val, '~id')
```

Valori ID personalizzati per le proprietà dei nodi e delle relazioni

A partire dal rilascio 1.2.0.2 del motore, Neptune ha esteso la specifica openCypher in modo che ora sia possibile specificare i valori `id` per i nodi e le relazioni nelle clausole `CREATE`, `MERGE` e `MATCH`. Ciò consente di assegnare stringhe intuitive anziché UUID generati dal sistema per identificare nodi e relazioni.

> **Warning**
>
> Questa estensione della specifica openCypher è incompatibile con le versioni precedenti, perché `~id` ora è considerato un nome di proprietà riservato. Se si utilizza già `~id` come proprietà nei dati e nelle query, è necessario migrare la proprietà esistente a una nuova chiave di proprietà e rimuovere quella precedente. Per informazioni, consulta Cosa fare se attualmente si utilizza `~id` come proprietà.

Ecco un esempio che mostra come creare nodi e relazioni con ID personalizzati:

```
CREATE (n {`~id`: 'fromNode', name: 'john'})
  -[:knows {`~id`: 'john-knows->jim', since: 2020}]
  ->(m {`~id`: 'toNode', name: 'jim'})
```

Se si tenta di creare un ID personalizzato già in uso, Neptune genera un errore `DuplicateDataException`.

Ecco un esempio di utilizzo di un ID personalizzato in una clausola `MATCH`:

```
MATCH (n {`~id`: 'id1'})
RETURN n
```

Ecco un esempio di utilizzo di ID personalizzati in una clausola `MERGE`:

```
MATCH (n {name: 'john'}), (m {name: 'jim'})
MERGE (n)-[r {`~id`: 'john->jim'}]->(m)
RETURN r
```

Cosa fare se attualmente si utilizza **`~id`** come proprietà

Con il rilascio 1.2.0.2 del motore, la chiave `~id` nelle clausole openCypher viene ora trattata come `id` anziché come proprietà. Ciò significa che se si dispone di una proprietà denominata `~id`, accedervi diventa impossibile.

Se si utilizza una proprietà `~id`, prima di aggiornare il motore al rilascio `1.2.0.2` o superiore è necessario migrare la proprietà `~id` esistente a una nuova chiave di proprietà e quindi rimuovere la proprietà `~id`. Ad esempio, la query seguente:

- Crea una nuova proprietà denominata 'newId' per tutti i nodi,
- copia il valore della proprietà '~id' nella proprietà 'newId'
- e rimuove la proprietà '~id' dai dati

```
MATCH (n)
WHERE exists(n.`~id`)
SET n.newId = n.`~id`
REMOVE n.`~id`
```

La stessa cosa deve essere fatta per tutte le relazioni nei dati che hanno una proprietà `~id`.

È inoltre necessario modificare tutte le query utilizzate che fanno riferimento a una proprietà `~id`. Ad esempio, questa query:

```
MATCH (n)
WHERE n.`~id` = 'some-value'
RETURN n
```

...verrà modificata in questa:

```
MATCH (n)
WHERE n.newId = 'some-value'
RETURN n
```

Altre differenze tra Neptune openCypher e Cypher

- Neptune supporta solo connessioni TCP per il protocollo Bolt. WebSocketsle connessioni per Bolt non sono supportate.
- Neptune openCypher rimuove gli spazi bianchi come definito da Unicode nelle funzioni `trim()`, `ltrim()` e `rtrim()`.
- In Neptune openCypher, `tostring(`double`)` non passa automaticamente alla notazione E per valori grandi di double.
- Sebbene openCypher CREATE non crei proprietà multivalore, queste possono esistere nei dati creati utilizzando Gremlin. Se Neptune openCypher incontra una proprietà multivalore, uno dei valori viene scelto arbitrariamente, creando un risultato non deterministico.

Modello di dati a grafo di Neptune

L'unità di base dei dati del grafo di Amazon Neptune è un elemento a quattro posizioni (quadrupla), simile a una quadrupla RDF (Resource Description Framework). Di seguito sono riportate le quattro posizioni di una quadrupla di Neptune:

- `subject  (S)`
- `predicate  (P)`
- `object  (O)`
- `graph  (G)`

Ogni quad è una dichiarazione che crea un'asserzione relativa a una o più risorse. Una dichiarazione può asserire l'esistenza di una relazione tra due risorse o può collegare una proprietà (coppia chiave-valore) a una risorsa. Il valore del predicato quad può essere considerato generalmente come il verbo dell'istruzione. Descrive il tipo di relazione o proprietà definita. L'oggetto è la destinazione della relazione o il valore della proprietà. Di seguito vengono mostrati gli esempi:

- Una relazione tra due vertici può essere rappresentata archiviando l'identificatore del vertice di origine nella posizione `S`, l'identificatore del vertice di destinazione nella posizione `O` e l'etichetta edge nella posizione `P`.
- Una proprietà può essere rappresentata archiviando l'identificatore dell'elemento nella posizione `S`, la chiave di proprietà nella posizione `P` e il valore della proprietà nella posizione `O`.

La posizione del grafico `G` viene utilizzata in modo diverso nei vari stack. Per i dati RDF in Neptune, la posizione `G` contiene un identificatore di grafo nominato. Per i grafici delle proprietà in Gremlin, viene utilizzato per archiviare il valore dell'ID edge nel caso di un edge. In tutti gli altri casi, come impostazione predefinita è un valore fisso.

Un set di dichiarazioni quad con identificatori di risorsa condivisi crea un grafico.

Dizionario dei valori rivolti all'utente

Neptune non archivia la maggior parte dei valori rivolti all'utente direttamente nei vari indici che gestisce. Li archivia invece separatamente in un dizionario e li sostituisce negli indici con identificatori a 8 byte.

- Tutti i valori rivolti all'utente che vengono inseriti negli indici `S`, `P` o `G` vengono archiviati nel dizionario in questo modo.
- Nell'indice 0, i valori numerici vengono archiviati direttamente nell'indice (in linea). Sono inclusi i valori `date` e `datetime` (rappresentati come millisecondi dall'epoca).
- Tutti gli altri valori rivolti all'utente che vengono inseriti nell'indice 0 vengono archiviati nel dizionario e rappresentati nell'indice da ID.

Il dizionario contiene una mappatura diretta dei valori rivolti all'utente con ID a 8 byte in un indice `value_to_id`.

Archivia la mappatura inversa degli ID a 8 byte rispetto ai valori in uno dei due indici, a seconda della dimensione dei valori:

- Un indice `id_to_value` mappa gli ID ai valori rivolti all'utente inferiori a 767 byte dopo la codifica interna.
- Un indice `id_to_blob` mappa gli ID ai valori rivolti all'utente più grandi.

In che modo le dichiarazioni vengono indicizzate in Neptune

Quando esegui una query su un grafico di quad, per ogni posizione quad puoi specificare o meno un vincolo di valore. La query restituisce tutti i quad che corrispondono ai vincoli di valore specificati.

Neptune utilizza gli indici per risolvere le query. Nel documento del 2005, Optimized Index Structures for Querying RDF from the Web (Strutture di indice ottimizzate per query su RDF dal Web) Andreas Harth e Stefan Decker hanno osservato che ci sono 16 (2^4) possibili modelli di accesso per le posizioni a quattro quad. Puoi eseguire query su tutti i 16 modelli in modo efficiente senza dover scansionare e filtrare utilizzando sei indici di istruzione quad. Ogni indice di istruzione quad utilizza una chiave composta dai quattro valori di posizione concatenati in un ordine diverso.

```
     Access Pattern                                        Index key order
-------------------------------------------------------    ---------------
 1.  ????  (No constraints; returns every quad)               SPOG
 2.  SPOG  (Every position is constrained)                    SPOG
 3.  SPO?  (S, P, and O are constrained; G is not)            SPOG
 4.  SP??  (S and P are constrained; O and G are not)         SPOG
 5.  S???  (S is constrained; P, O, and G are not)            SPOG
 6.  S??G  (S and G are constrained; P and O are not)         SPOG
```

```
 7.  ?POG  (P, O, and G are constrained; S is not)          POGS
 8.  ?PO?  (P and O are constrained; S and G are not)       POGS
 9.  ?P??  (P is constrained; S, O, and G are not)          POGS

10.  ?P?G  (P and G are constrained; S and O are not)       GPSO
11.  SP?G  (S, P, and G are constrained; O is not)          GPSO
12.  ???G  (G is constrained; S, P, and O are not)          GPSO

13.  S?OG  (S, O, and G are constrained; P is not)          OGSP
14.  ??OG  (O and G are constrained; S and P are not)       OGSP
15.  ??O?  (O is constrained; S, P, and G are not)          OGSP

16.  S?O?  (S and O are constrained; P and G are not)       OSGP
```

Per impostazione predefinita Neptune crea e gestisce solo tre di questi sei indici:

- `SPOG` – Utilizza una chiave composta da `Subject + Predicate + Object + Graph`.
- `POGS` – Utilizza una chiave composta da `Predicate + Object + Graph + Subject`.
- `GPSO` – Utilizza una chiave composta da `Graph + Predicate + Subject + Object`.

Questi tre indici gestiscono molti dei modelli di accesso più comuni. Mantenere solo tre indici di istruzione completi anziché sei riduce notevolmente le risorse necessarie per supportare l'accesso rapido senza scansionare e filtrare. Ad esempio, l'indice `SPOG` consente una ricerca efficiente ogni volta che viene associato un prefisso delle posizioni, ad esempio il vertice o il vertice e l'identificatore di proprietà. L'indice `POGS` consente un accesso efficiente quando solo l'etichetta della proprietà o di edge archiviata nella posizione `P` è associata.

L'interfaccia API di basso livello per dichiarazioni di ricerca richiede un modello di dichiarazione in cui alcune posizioni sono note e le restanti vengono lasciate per il rilevamento con ricerca per indice. Componendo le posizioni note in un prefisso chiave in base all'ordine della chiave di indicizzazione per uno degli indici di dichiarazione, Neptune esegue una scansione dell'intervallo per recuperare tutte le dichiarazioni corrispondenti alle posizioni note.

Uno degli indici di dichiarazione che Neptune non crea per impostazione predefinita, tuttavia, è un indice `OSGP` trasversale inverso, che può raccogliere predicati tra oggetti e soggetti. Al contrario, per impostazione predefinita Neptune tiene traccia di predicati distinti in un indice separato che utilizza per eseguire una scansione di unione di `{all P x POGS}`. Quando utilizzi Gremlin, un predicato corrisponde ad una proprietà o un'etichetta di edge.

Tuttavia, se il numero di predicati distinti in un grafico diventa di elevate dimensioni, la strategia di accesso predefinita di Neptune può diventare inefficiente. In Gremlin, ad esempio una fase `in()` in cui non vengono fornite etichette edge o qualsiasi fase che utilizza `in()` internamente, come `both()` o `drop()`, può diventare piuttosto inefficiente.

Abilitazione della creazione dell'indice OSPG utilizzando la modalità di laboratorio

Se il modello di dati crea un numero elevato di predicati distinti, è possibile che si verifichino prestazioni ridotte e costi operativi più elevati che possono essere notevolmente migliorati utilizzando la modalità laboratorio per abilitare l'indice OSGP, oltre ai tre indici che Neptune mantiene per impostazione predefinita.

> **Note**
>
> Questa funzionalità è disponibile a partire dal rilascio 1.0.1.0.200463.0 del motore Neptune.

L'abilitazione dell'indice OSGP può avere alcuni aspetti negativi:

- La velocità di inserimento potrebbe rallentare fino al 23%.
- Lo storage aumenta fino al 20%.
- Le query di lettura che interessano tutti gli indici allo stesso modo (il che è piuttosto raro) potrebbero presentare aumenti delle latenze.

In generale, tuttavia, vale la pena abilitare l'indice OSPG per i cluster database con un numero elevato di predicati distinti. Le ricerche basate su oggetti diventano altamente efficienti (ad esempio, trovare tutti gli archi in entrata su un vertice, o tutti i soggetti collegati a un determinato oggetto) e, di conseguenza, l'eliminazione dei vertici diventa molto più efficiente.

> **Important**
>
> È possibile abilitare l'indice OSGP solo in un cluster database vuoto, prima di caricarvi i dati.

Dichiarazioni Gremlin nel modello di dati di Neptune

I dati del grafo delle proprietà Gremlin sono espressi nel modello SPOG utilizzando tre classi di dichiarazioni, vale a dire:

- Dichiarazioni dell'etichetta del vertice
- Dichiarazioni di edge
- Dichiarazioni di proprietà

Per una spiegazione di come vengono utilizzati nelle query Gremlin, vedi Introduzione al funzionamento delle query Gremlin in Neptune.

La cache di ricerca di Neptune può accelerare le query di lettura

Amazon Neptune implementa una cache di ricerca che utilizza l'SSD basato su NVMe dell'istanza `R5d` per migliorare le prestazioni di lettura per le query con ricerche frequenti e ripetitive di valori di proprietà o valori letterali RDF. La cache di ricerca memorizza temporaneamente questi valori nel volume SSD NVMe, dove è possibile accedervi rapidamente.

Questa funzionalità è disponibile a partire dal Motore Amazon Neptune versione 1.0.4.2.R2 (01/06/2021).

Le query di lettura che restituiscono le proprietà di un gran numero di vertici e archi, o di molte triple RDF, possono avere una latenza elevata se i valori delle proprietà o i valori letterali devono essere recuperati dai volumi di archiviazione del cluster anziché dalla memoria. Gli esempi includono le query di lettura di lunga durata che restituiscono un gran numero di nomi completi da un grafo di identità o di indirizzi IP da un grafo di rilevamento delle frodi. All'aumentare del numero di valori di proprietà o di valori letterali RDF restituiti dalla query, la memoria disponibile diminuisce e l'esecuzione delle query può peggiorare notevolmente.

Casi d'uso per la cache di ricerca di Neptune

La cache di ricerca è utile solo quando le query di lettura restituiscono le proprietà di un numero molto elevato di vertici e archi o di triple RDF.

Per ottimizzare le prestazioni delle query, Amazon Neptune utilizza il tipo di istanza `R5d` per creare una cache di grandi dimensioni per tali valori di proprietà o valori letterali. Il recupero dalla cache è quindi molto più rapido rispetto al recupero dai volumi di archiviazione del cluster.

Come regola generale, vale la pena abilitare la cache di ricerca solo se sono soddisfatte tutte e tre le seguenti condizioni:

- Si osserva un aumento della latenza nelle query di lettura.
- Si osserva anche un calo della `BufferCacheHitRatio` CloudWatch metrica durante l'esecuzione di query di lettura (vedi). Monitoraggio di Neptune tramite Amazon CloudWatch
- Le query di lettura impiegano molto tempo per materializzare i valori restituiti prima del rendering dei risultati (vedi l'esempio del profilo Gremlin riportato di seguito per determinare quanti valori di proprietà vengono materializzati per una query).

> **Note**
>
> Questa funzionalità è utile solo nello scenario specifico descritto sopra. Ad esempio, la cache di ricerca non aiuta affatto le query di aggregazione. A meno che non si eseguano query che potrebbero trarre vantaggio dalla cache di ricerca, non c'è motivo di utilizzare un tipo di istanza R5d anziché un tipo di istanza R5 equivalente e meno costoso.

Se si utilizza Gremlin, è possibile valutare i costi di materializzazione di una query con API Gremlin `profile`. In "Operazioni di indice" viene mostrato il numero di termini materializzati durante l'esecuzione:

```
Index Operations
Query execution:
    # of statement index ops: 3
    # of unique statement index ops: 3
    Duplication ratio: 1.0
    # of terms materialized: 5273
Serialization:
    # of statement index ops: 200
    # of unique statement index ops: 140
    Duplication ratio: 1.43
    # of terms materialized: 32693
```

Il numero di termini non numerici che vengono materializzati è direttamente proporzionale al numero di ricerche di termini che Neptune deve eseguire.

Utilizzo della cache di ricerca

La cache di ricerca è disponibile solo su un tipo di istanza R5d, dove è abilitata automaticamente per impostazione predefinita. Le istanze Neptune R5d hanno le stesse specifiche delle istanze R5, oltre a un massimo di 1,8 TB di archiviazione SSD locale basato su NVMe. Le cache di ricerca sono specifiche dell'istanza e i carichi di lavoro che ne traggono vantaggio possono essere indirizzati specificamente alle istanze R5d di un cluster Neptune, mentre altri carichi di lavoro possono essere indirizzati a R5 o ad altri tipi di istanze.

Per utilizzare la cache di ricerca su un'istanza Neptune, è sufficiente aggiornare l'istanza al tipo di istanza R5d. In questo modo, Neptune imposta automaticamente il parametro del cluster database neptune_lookup_cache su `'enabled'` e crea la cache di ricerca su quella particolare istanza. È quindi possibile utilizzare l'API Stato dell'istanza per confermare che la cache è stata abilitata.

Analogamente, per disabilitare la cache di ricerca su una determinata istanza, dimensionare l'istanza da un tipo di istanza R5d a un tipo di istanza R5 equivalente.

Quando viene avviata un'istanza R5d, la cache di ricerca è abilitata e in modalità di avvio a freddo, ovvero è vuota. Neptune verifica innanzitutto la presenza nella cache di ricerca di valori di proprietà o di valori letterali RDF durante l'elaborazione delle query e li aggiunge se non sono ancora presenti. In questo modo la cache si riscalda gradualmente.

Quando si indirizzano le query di lettura che richiedono ricerche di valori di proprietà o di valori letterali RDF a un'istanza di lettura R5d, le prestazioni di lettura diminuiscono leggermente durante il riscaldamento della cache. Quando la cache si riscalda, tuttavia, le prestazioni di lettura aumentano notevolmente e si può anche notare un calo dei costi di I/O legati alle ricerche che vengono effettuate sulla cache anziché sull'archiviazione del cluster. Migliora anche l'utilizzo della memoria.

Se l'istanza di scrittura è R5d, riscalda automaticamente la cache di ricerca a ogni operazione di scrittura. Questo approccio aumenta leggermente la latenza per le query di scrittura, ma riscalda la cache di ricerca in modo più efficiente. Quindi, se si indirizzano le query di lettura che richiedono ricerche di valori di proprietà o di valori letterali RDF all'istanza di scrittura, le prestazioni di lettura migliorano immediatamente, poiché i valori sono già stati memorizzati nella cache.

Inoltre, se si esegue lo strumento di caricamento in blocco su un'istanza di scrittura R5d, si può notare che le sue prestazioni sono leggermente ridotte a causa della cache.

Poiché la cache di ricerca è specifica per ogni nodo, la sostituzione dell'host ripristina la cache su un avvio a freddo.

È possibile disabilitare temporaneamente la cache di ricerca in tutte le istanze del cluster database impostando il parametro del cluster database neptune_lookup_cache su `'disabled'`. In generale, tuttavia, ha più senso disabilitare la cache in istanze specifiche dimensionandole dal tipo di istanza R5d al tipo R5.

Semantica delle transazioni in Neptune

Amazon Neptune è progettato per supportare carichi di lavoro OLTP (Online Transactional Processing) ad elevata simultaneità su grafi dei dati. La specifica W3C SPARQL Query Language for RDF e la documentazione di Apache TinkerPop Gremlin Graph Traversal Language non definiscono la semantica delle transazioni per l'elaborazione simultanea delle query. Poiché il supporto ACID e le garanzie sulle transazioni ben definite possono essere molto importanti, applichiamo una semantica rigorosa per evitare anomalie nei dati.

Questa sezione definisce queste semantiche e illustra come vengono applicate ad alcuni casi d'uso comuni in Neptune.

Argomenti

Definizione dei livelli di isolamento

La "I" in `ACID` sta per isolamento. Il grado di isolamento di una transazione determina quanto o quanto poco altre transazioni simultanee possono influire sui dati su cui opera.

Lo standard SQL:1992 ha creato un vocabolario per descrivere i livelli di isolamento. Definisce tre tipi di interazioni (che chiama fenomeni) che possono verificarsi tra due transazioni simultanee, `Tx1` e `Tx2`:

- `Dirty read`: si verifica quando `Tx1` modifica un elemento e quindi `Tx2` legge quell'elemento prima che `Tx1` abbia completato la modifica. Pertanto, se `Tx1` non riesce mai a eseguire il commit della modifica o a eseguire il rollback, `Tx2` ha letto un valore che non è mai entrato nel database.
- `Non-repeatable read`: si verifica quando `Tx1` legge un elemento, quindi `Tx2` modifica o elimina quell'elemento e completa la modifica, quindi `Tx1` tenta di rileggere l'elemento. `Tx1` ora legge un valore diverso rispetto a prima o rileva che l'elemento non esiste più.
- `Phantom read`: si verifica quando `Tx1` legge un insieme di elementi che soddisfano un criterio di ricerca, quindi `Tx2` aggiunge un nuovo elemento che soddisfa il criterio di ricerca e quindi `Tx1` ripete la ricerca. `Tx1` ora ottiene un insieme di elementi diverso rispetto a prima.

Ognuno di questi tre tipi di interazione può causare incoerenze nei dati risultanti in un database.

Lo standard SQL:1992 ha definito quattro livelli di isolamento che presentano garanzie diverse per quanto riguarda i tre tipi di interazione e le incoerenze che possono produrre. A tutti e quattro i livelli, è possibile garantire che una transazione venga eseguita completamente o non venga eseguita affatto:

- `READ UNCOMMITTED`: consente tutti e tre i tipi di interazione (ovvero letture dirty, letture non ripetibili e letture fantasma).
- `READ COMMITTED`: le letture dirty non sono possibili, ma le letture non ripetibili e le letture fantasma lo sono.
- `REPEATABLE READ`: non sono possibili né le letture dirty né le letture non ripetibili, ma sono ancora possibili le letture fantasma.
- `SERIALIZABLE`: nessuno dei tre tipi di interazione è possibile.

Multiversion Concurrency Control (MVCC) consente un altro tipo di isolamento, ovvero l'isolamento SNAPSHOT . Ciò garantisce che una transazione funzioni su uno snapshot di dati esistente all'inizio della transazione e che nessun'altra transazione possa modificare tale snapshot.

Livelli di isolamento della transazione in Neptune

Amazon Neptune implementa diversi livelli di isolamento della transazione per query di sola lettura e query di mutazione. Le query SPARQL e Gremlin sono classificate come di sola lettura o di mutazione in base ai seguenti criteri:

- In SPARQL, esiste una chiara distinzione tra query di lettura (`SELECT`, `ASK`, `CONSTRUCT` e `DESCRIBE` come definito nella specifica SPARQL 1.1 Query Language) e query di mutazione (`INSERT` e `DELETE` come definito nella specifica SPARQL 1.1 Update).

  Tenere presente che Neptune tratta più query di mutazione inviate insieme (ad esempio, in un messaggio `POST` separato da punto e virgola) come una singola transazione. Sono garantite per avere successo o fallire come unità atomica e, in caso di errore, le modifiche parziali vengono ripristinate.
- Tuttavia, in Gremlin, Neptune classifica una query come di sola lettura o di mutazione a seconda che contenga eventuali passaggi del percorso di query, ad esempio `addE()`, `addV()`, `property()` o `drop()` che manipola i dati. Se la query contiene una tale fase di percorso, viene classificata ed eseguita come query di mutazione.

In Gremlin è anche possibile utilizzare sessioni di standing. Per ulteriori informazioni, consulta Sessioni basate su script Gremlin. In queste sessioni, tutte le query, incluse le query di sola lettura, vengono eseguite con lo stesso isolamento delle query di mutazione sull'endpoint di scrittura.

Utilizzando sessioni di lettura-scrittura in openCypher, tutte le query, incluse le query di sola lettura, vengono eseguite con lo stesso isolamento delle query di mutazione sull'endpoint di scrittura.

Argomenti

- Isolamento delle query di sola lettura in Neptune
- Isolamento delle query di mutazione in Neptune
- Risoluzione dei conflitti tramite timeout di attesa di blocco
- Blocchi di intervalli e falsi conflitti

Isolamento delle query di sola lettura in Neptune

Neptune valuta le query di sola lettura nella semantica di isolamento degli snapshot. Ciò significa che una query di sola lettura opera logicamente su uno snapshot coerente del database acquisito all'inizio della valutazione della query. Neptune può quindi garantire che nessuno dei seguenti fenomeni si verificherà:

- `Dirty reads`: le query di sola lettura in Neptune non vedranno mai i dati non vincolati di una transazione simultanea.
- `Non-repeatable reads`: una transazione di sola lettura che legge gli stessi dati più di una volta restituirà sempre gli stessi valori.
- `Phantom reads`: una transazione di sola lettura non leggerà mai i dati aggiunti dopo l'inizio della transazione.

Poiché l'isolamento degli snapshot viene ottenuto utilizzando Multiversion Concurrency Control (MVCC), le query di sola lettura non devono bloccare i dati e pertanto non bloccano le query di mutazione.

Le repliche di lettura accettano solo query di sola lettura, pertanto tutte le query sulle repliche di lettura vengono eseguite nella semantica di isolamento `SNAPSHOT`.

L'unica considerazione aggiuntiva quando si esegue una query su una replica di lettura è che può verificarsi un piccolo ritardo di replica tra le repliche di scrittura e di lettura. Ciò significa che un

aggiornamento effettuato su una replica di scrittura potrebbe richiedere poco tempo per essere propagato alla replica di lettura da cui si sta leggendo. Il tempo di replica effettivo dipende dal carico di scrittura sull'istanza primaria. L'architettura Neptune supporta la replica a bassa latenza e il ritardo di replica è strumentato in un parametro Amazon. CloudWatch

Tuttavia, a causa del livello di isolamento `SNAPSHOT`, le query di lettura visualizzano sempre uno stato coerente del database, anche se non è il più recente.

Nei casi in cui è richiesta una forte garanzia che una query osservi il risultato di un aggiornamento precedente, invia la query all'endpoint di scrittura stesso anziché a una replica di lettura.

Isolamento delle query di mutazione in Neptune

Le letture effettuate come parte delle query di mutazione vengono eseguite sotto l'isolamento della transazione `READ COMMITTED`, il che esclude la possibilità di letture modificate. Superando le normali garanzie fornite per l'isolamento della transazione `READ COMMITTED`, Neptune fornisce la garanzia solida che né le letture `NON-REPEATABLE` né le letture `PHANTOM` possono verificarsi.

Queste garanzie solide si ottengono bloccando record e intervalli di record durante la lettura dei dati. Questo impedisce alle transazioni simultanee di effettuare inserzioni o eliminazioni negli intervalli di indice dopo che sono state lette, garantendo così letture ripetibili.

> **Note**
>
> Tuttavia, una transazione di mutazione simultanea `Tx2` potrebbe iniziare dopo l'avvio della transazione di mutazione `Tx1` e potrebbe eseguire il commit di una modifica prima che `Tx1` abbia bloccato i dati per leggerli. In questo caso, `Tx1` vedrebbe la modifica di `Tx2` come se `Tx2` fosse stata completata prima dell'avvio di `Tx1`. Poiché ciò si applica solo alle modifiche di cui è stato eseguito il commit, un `dirty read` non può mai verificarsi.

Per comprendere il meccanismo di blocco utilizzato da Neptune per le query di mutazione, è utile innanzitutto comprendere i dettagli di Neptune Il modello di dati Graph e Strategia di indicizzazione. Neptune gestisce i dati utilizzando tre indici, vale a dire, `SPOG`, `POGS` e `GPSO`.

Per ottenere letture ripetibili per il livello di transazione `READ COMMITTED`, Neptune accetta blocchi di intervalli nell'indice utilizzato. Ad esempio, se una query di mutazione legge tutte le proprietà e gli edge in uscita di un vertice denominato `person1`, il nodo blocca l'intero intervallo definito dal prefisso `S=person1` nell'indice `SPOG` prima di leggere i dati.

Lo stesso meccanismo si applica quando si utilizzano altri indici. Ad esempio, quando una transazione di mutazione cerca tutte le coppie di vertici origine-destinazione per una determinata etichetta edge utilizzando l'indice `POGS`, l'intervallo per l'etichetta edge nella posizione `P` sarebbe bloccato. Qualsiasi transazione simultanea, a prescindere che sia una query di sola lettura o di mutazione, può comunque eseguire letture all'interno dell'intervallo bloccato. Tuttavia, qualsiasi mutazione che implichi l'inserimento o l'eliminazione di nuovi record nell'intervallo di prefissi bloccato richiederebbe un blocco esclusivo e verrebbe impedita.

In altre parole, quando un intervallo dell'indice è stato letto da una transazione di mutazione, c'è una forte garanzia che questo intervallo non verrà modificato da qualsiasi transazione simultanea fino al termine della transazione di lettura. Ciò garantisce che non si verificherà alcuna operazione `non-repeatable reads`.

Risoluzione dei conflitti tramite timeout di attesa di blocco

Se una seconda transazione cerca di modificare un record in un intervallo bloccato da una prima transazione, Neptune rileva immediatamente il conflitto e blocca la seconda transazione.

Se non viene rilevato alcun deadlock delle dipendenze, Neptune applica automaticamente un meccanismo di timeout di attesa del blocco, in cui la transazione bloccata attende la transazione che contiene il blocco fino a 60 secondi per terminare e rilasciare il blocco.

- Se il timeout di attesa del blocco scade prima che il blocco venga rilasciato, viene eseguito il rollback della transazione bloccata.
- Se il blocco viene rilasciato entro il timeout di attesa del blocco, la seconda transazione viene sbloccata e può terminare senza che sia necessario riprovare.

Tuttavia, se Neptune rileva un deadlock delle dipendenze tra le due transazioni, non è possibile eseguire la riconciliazione automatica del conflitto. In questo caso, Neptune annulla immediatamente ed esegue il rollback di una delle due transazioni senza avviare un timeout di attesa del blocco. Neptune fa il massimo sforzo per eseguire il rollback della transazione con il minor numero di record inseriti o eliminati.

Blocchi di intervalli e falsi conflitti

Neptune accetta i blocchi di intervalli usando i blocchi gap. Un blocco gap è un blocco su un gap tra i record dell'indice o un blocco sul gap prima del primo o dopo l'ultimo record dell'indice.

Neptune utilizza una cosiddetta tabella di dizionari per associare valori ID numerici a valori letterali di stringa specifici. Ecco un esempio di stato di un dizionario Neptune: tabella:

Stringa	ID
type (tipo)	1
default_graph	2
person_3	3
person_1	5
knows	6
person_2	7
age	8
edge_1	9
lives_in	10
New York	11
Person	12
Place	13
edge_2	14

Le stringhe sopra riportate appartengono a un modello di grafo delle proprietà, ma i concetti si applicano allo stesso modo anche a tutti i modelli di grafo RDF.

Lo stato corrispondente dell'indice SPOG (Subject-Predicate-Object_Graph) è mostrato in basso a sinistra. A destra, vengono mostrate le stringhe corrispondenti, per aiutare a capire il significato dei dati dell'indice.

S (ID)	P (ID)	O (ID)	G (ID)		S (stringa)	P (stringa)	O (stringa)	G (stringa)
3	1	12	2		person_3	type (tipo)	Person	default_graph
5	1	12	2		person_1	type (tipo)	Person	default_graph
5	6	3	9		person_1	knows	person_3	edge_1
5	8	40	2		person_1	age	40	default_graph
5	10	11	14		person_1	lives_in	New York	edge_2
7	1	12	2		person_2	type (tipo)	Person	default_graph
11	1	13	2		New York	type (tipo)	Place	default_graph

Ora, se una query di mutazione legge tutte le proprietà e gli archi in uscita di un vertice denominato `person_1`, il nodo blocca l'intero intervallo definito dal prefisso `S=person_1` nell'indice SPOG prima di leggere i dati. Il blocco dell'intervallo inserirà blocchi gap su tutti i record corrispondenti e sul primo record che non corrisponde. I record corrispondenti verranno bloccati, mentre quelli non corrispondenti non verranno bloccati. Neptune inserirà i blocchi gap come segue:

- `5 1 12 2` (gap 1)
- `5 6 3 9` (gap 2)
- `5 8 40 2` (gap 3)
- `5 10 11 14` (gap 4)
- `7 1 12 2` (gap 5)

In questo modo vengono bloccati i seguenti record:

- `5 1 12 2`
- `5 6 3 9`
- `5 8 40 2`
- `5 10 11 14`

In questo stato, le seguenti operazioni sono legittimamente bloccate:

- Inserimento di una nuova proprietà o arco per `S=person_1`. Una nuova proprietà diversa da `type` o un nuovo arco devono essere inseriti nel gap 2, nel gap 3, nel gap 4 o nel gap 5, che sono tutti bloccati.
- Eliminazione di qualsiasi record esistente.

Allo stesso tempo, alcune operazioni simultanee verranno bloccate in modo errato (generando falsi conflitti):

- Qualsiasi inserimento di proprietà o archi per `S=person_3` viene bloccato perché deve essere inserito nel gap 1.
- Qualsiasi nuovo inserimento di vertici a cui viene assegnato un ID compreso tra 3 e 5 verrà bloccato perché deve essere inserito nel gap 1.
- Qualsiasi nuovo inserimento di vertici a cui viene assegnato un ID compreso tra 5 e 7 verrà bloccato perché deve essere inserito nel gap 5.

I blocchi gap non sono sufficientemente precisi per bloccare il gap per un predicato specifico (ad esempio, per bloccare gap5 per il predicato `S=5`).

I blocchi di intervallo vengono posizionati solo nell'indice in cui avviene la lettura. Nel caso precedente, i record sono bloccati solo nell'indice SPOG, non in POGS o GPSO. Le letture di una query possono essere eseguite su tutti gli indici a seconda dei modelli di accesso, che possono essere elencati utilizzando le API `explain` (per Sparql e per Gremlin).

> **Note**
>
> I blocchi gap possono anche essere utilizzati per aggiornamenti simultanei sicuri sugli indici sottostanti, il che può anche portare a falsi conflitti. Questi blocchi gap vengono posizionati

indipendentemente dal livello di isolamento o dalle operazioni di lettura eseguite dalla transazione.

I falsi conflitti possono verificarsi non solo quando le transazioni simultanee entrano in conflitto a causa di blocchi gap, ma anche in alcuni casi quando una transazione viene ritentata dopo qualsiasi tipo di errore. Se il rollback attivato dall'errore è ancora in corso e i blocchi precedentemente utilizzati per la transazione non sono ancora stati completamente rilasciati, il nuovo tentativo genererà un falso conflitto e avrà esito negativo.

Con un carico elevato, in genere è possibile che il 3-4% delle query di scrittura non riesca a causa di falsi conflitti. Per un client esterno, tali falsi conflitti sono difficili da prevedere e devono essere gestiti mediante nuovi tentativi.

Esempi di semantica delle transazioni Neptune

I seguenti esempi illustrano diversi casi d'uso per la semantica delle transazioni in Amazon Neptune.

Argomenti

Esempio 1: Inserimento di una proprietà solo se non esiste

Supponi di volerti assicurare che una proprietà sia impostata solo una volta. Ad esempio, supponi che più query stiano tentando di assegnare a una persona un punteggio di credito simultaneamente. Desideri che venga inserita una sola un'istanza della proprietà e che le altre query non vadano a buon fine perché la proprietà è già stata impostata.

```
# GREMLIN:
g.V('person1').hasLabel('Person').coalesce(has('creditScore'), property('creditScore',
 'AAA+'))

# SPARQL:
```

```
INSERT { :person1 :creditScore "AAA+" .}
WHERE  { :person1 rdf:type :Person .
         FILTER NOT EXISTS { :person1 :creditScore ?o .} }
```

La fase `property()` di Gremlin inserisce una proprietà con la chiave e il valore forniti. La fase `coalesce()` esegue il primo argomento nella prima fase e se non va a buon fine, esegue la seconda fase:

Prima di inserire il valore per la proprietà `creditScore` per un determinato vertice `person1`, una transazione deve cercare di leggere il valore `creditScore` possibilmente inesistente per `person1`. Questa lettura tentata blocca l'intervallo SP per S=`person1` e P=`creditScore` nell'indice SPOG in cui il valore `creditScore` esiste o verrà scritto.

L'uso di questo blocco di intervallo impedisce a qualsiasi transazione simultanea di inserire un valore `creditScore` simultaneamente. Quando sono presenti più transazioni parallele, solo una alla volta può aggiornare il valore. Ciò esclude l'anomalia di creazione di più proprietà `creditScore`.

Esempio 2: Affermare che il valore di una proprietà è univoco a livello globale

Supponi di voler inserire una persona con un numero di previdenza sociale come una chiave primaria. Desideri che la query di mutazione garantisca che, a livello globale, nessun altro nel database abbia lo stesso numero di previdenza sociale:

```
# GREMLIN:
g.V().has('ssn', 123456789).fold()
  .coalesce(__.unfold(),
            __.addV('Person').property('name', 'John Doe').property('ssn', 123456789'))

# SPARQL:
INSERT { :person1 rdf:type :Person .
         :person1 :name "John Doe" .
         :person1 :ssn 123456789 .}
WHERE  { FILTER NOT EXISTS { ?person :ssn 123456789 } }
```

Questo esempio è simile a quello precedente. La differenza principale è che il blocco dell'intervallo viene eseguito sull'indice POGS anziché sull'indice SPOG.

La transazione che esegue la query deve leggere il modello, `?person :ssn 123456789`, in cui le posizioni P e 0 sono vincolate. Il blocco dell'intervallo viene eseguito sull'indice POGS per P=`ssn` e 0=123456789.

- Se il modello esiste, non viene eseguita alcuna operazione.
- In caso contrario, il blocco impedisce a qualsiasi transazione simultanea di inserire anche tale numero di previdenza sociale

Esempio 3: Modifica di una proprietà se un'altra proprietà ha un valore specificato

Supponi che vari eventi in un gioco spostino una persona dal livello 1 al livello 2 e che venga loro assegnata una nuova proprietà `level2Score` impostata su zero. Occorre accertarsi che più istanze simultanee di tale transazione non siano in grado di creare più istanze della proprietà score a livello due. Le query in Gremlin e SPARQL potrebbero essere simili alle seguenti.

```
# GREMLIN:
g.V('person1').hasLabel('Person').has('level', 1)
 .property('level2Score', 0)
 .property(Cardinality.single, 'level', 2)

# SPARQL:
DELETE { :person1 :level 1 .}
INSERT { :person1 :level2Score 0 .
         :person1 :level 2 .}
WHERE  { :person1 rdf:type :Person .
         :person1 :level 1 .}
```

In Gremlin, quando si specifica `Cardinality.single`, la fase `property()` aggiunge una nuova proprietà o sostituisce un valore di proprietà esistente con il nuovo valore specificato.

Qualsiasi aggiornamento a un valore di proprietà, ad esempio incrementando `level` da 1 a 2, viene implementato come un'eliminazione del record corrente e l'inserimento di un nuovo record con il nuovo valore di proprietà. In questo caso, il record con il numero di livello 1 viene eliminato e un record con il numero di livello 2 viene reinserito.

Affinché la transazione sia in grado di aggiungere `level2Score` e aggiornare `level` da 1 a 2, è necessario innanzitutto verificare che il valore `level` sia attualmente pari a 1. In questo modo, esegue un blocco dell'intervallo sul prefisso SPO per `S=person1`, `P=level` e `O=1` nell'indice SPOG. Questo blocco impedisce alle transazioni simultanee di eliminare la tripla versione 1 e, di conseguenza, non possono verificarsi aggiornamenti simultanei in conflitto.

Esempio 4: Sostituzione di una proprietà esistente

Alcuni eventi potrebbero aggiornare il punteggio di credito di una persona a un nuovo valore (qui BBB). Tuttavia, vuoi essere certo che gli eventi simultanei di tale tipo non possano creare più proprietà del punteggio di credito per una persona.

```
# GREMLIN:
g.V('person1').hasLabel('Person')
 .sideEffect(properties('creditScore').drop())
 .property('creditScore', 'BBB')

# SPARQL:
DELETE { :person1 :creditScore ?o .}
INSERT { :person1 :creditScore "BBB" .}
WHERE  { :person1 rdf:type :Person .
         :person1 :creditScore ?o .}
```

Questo caso è simile all'esempio 3, tranne per il fatto che anziché bloccare il prefisso SPO, Neptune blocca il prefisso SP solo con S=`person1` e P=`creditScore`. Questo impedisce alle transazioni simultanee di inserire o eliminare triple con la proprietà `creditScore` per il soggetto `person1`.

Esempio 5: Evitare proprietà o archi pendenti

L'aggiornamento di un'entità non deve lasciare un elemento pendente, ovvero una proprietà o edge associata a un'entità per cui non è definito un tipo. Questo è un problema solo in SPARQL, perché Gremlin ha vincoli integrati per evitare elementi pendenti.

```
# SPARQL:
tx1: INSERT { :person1 :age 23 } WHERE { :person1 rdf:type :Person }
tx2: DELETE { :person1 ?p ?o }
```

La query `INSERT` deve leggere e bloccare il prefisso SPO con S=`person1`, P=`rdf:type` e O=`Person` nell'indice SPOG. Il blocco impedisce che la query `DELETE` vada a buon fine in parallelo.

Nel conflitto tra la query `DELETE` che cerca di eliminare il record `:person1 rdf:type :Person` e la query `INSERT` che legge il record e crea un blocco dell'intervallo sul suo SPO nell'indice SPOG, sono possibili i seguenti risultati:

- Se la query `INSERT` esegue il commit prima che la query `DELETE` legga ed elimini tutti i record per `:person1`, `:person1` viene rimosso completamente dal database, incluso il nuovo record inserito.

- Se la query `DELETE` esegue il commit prima che la query `INSERT` cerchi di leggere il record `:person1 rdf:type :Person`, la lettura osserva la modifica sottoposta a commit. Ovvero, non trova alcun record `:person1 rdf:type :Person` e quindi diventa un no-op.
- Se la query `INSERT` legge prima che lo faccia la query `DELETE`, la tripla `:person1 rdf:type :Person` viene bloccata e la query `DELETE` viene bloccata fino al commit della query INSERT, come nel primo caso in precedenza.
- Se `DELETE` legge prima della query `INSERT` e la query `INSERT` cerca di leggere e di eseguire un blocco sul prefisso `SPO` per il record, viene rilevato un conflitto. Questo perché la tripla è stata contrassegnata per la rimozione e `INSERT` non va a buon fine.

In tutte queste possibili sequenze di eventi, non viene creato alcun edge pendente.

Gestione di eccezioni e nuovi tentativi

Quando le transazioni vengono annullate a causa di conflitti non risolvibili o timeout di attesa del blocco, Amazon Neptune risponde con `ConcurrentModificationException`. Per ulteriori informazioni, consulta Codici di errore del motore. Come best practice, i client devono sempre catturare e gestire queste eccezioni.

In molti casi, quando il numero di istanze di `ConcurrentModificationException` è basso, un meccanismo di tentativi basato su backoff esponenziale è un metodo di gestione ideale. In questo approccio di tentativi, il numero massimo di tentativi e il tempo di attesa dipende generalmente dalle dimensioni massime e dalla durata delle transazioni.

Tuttavia, se l'applicazione dispone di carichi di lavoro di aggiornamento a elevata simultaneità e si osserva un numero elevato di eventi `ConcurrentModificationException`, puoi modificare l'applicazione per ridurre il numero di modifiche simultanee in conflitto.

Ad esempio, considera un'applicazione che effettua aggiornamenti frequenti a un set di vertici e utilizza più thread simultanei per questi aggiornamenti per ottimizzare il throughput di scrittura. Se ogni thread esegue continuamente query che aggiornano una o più proprietà del nodo, gli aggiornamenti simultanei dello stesso nodo possono produrre `ConcurrentModificationException`. Questo a sua volta può compromettere le prestazioni di scrittura.

È possibile ridurre notevolmente la probabilità di tali collisioni se è possibile serializzare gli aggiornamenti che potrebbero essere in conflitto tra loro. Ad esempio, se puoi garantire che tutte le query di aggiornamento per un determinato nodo vengano eseguite sullo stesso thread (magari

utilizzando un'assegnazione basata su hash), puoi essere certo che verranno eseguite una dopo l'altra anziché simultaneamente. Anche se è comunque possibile che un blocco dell'intervallo eseguito su un nodo adiacente possa causare una `ConcurrentModificationException`, gli aggiornamenti simultanei allo stesso nodo vengono eliminati.

Cluster e istanze database di Amazon Neptune

Un cluster database Amazon Neptune gestisce l'accesso ai dati tramite query. Un cluster è composto da:

- Un'istanza database primaria.
- Fino a 15 istanze database di replica di lettura.

Tutte le istanze di un cluster condividono lo stesso volume di archiviazione gestito sottostante, progettato per garantire affidabilità e disponibilità elevata.

La connessione alle istanze database del cluster database avviene tramite endpoint Neptune.

Istanza database primaria in un cluster database Neptune

L'istanza database primaria coordina tutte le operazioni di scrittura sul volume di archiviazione sottostante del cluster database. Supporta anche le operazioni di lettura.

Può esserci solo un'istanza database primaria in un cluster database Neptune. Se l'istanza primaria non è più disponibile, Neptune esegue automaticamente il failover su una delle istanze di replica di lettura con una priorità che è possibile specificare.

Istanze database di replica di lettura in un cluster database Neptune

Dopo avere creato l'istanza primaria per un cluster database, puoi creare fino a 15 istanze di replica di lettura nel cluster database, al fine di supportare le query di sola lettura.

Le istanze database di replica di lettura Neptune funzionano bene per il dimensionamento della capacità di lettura perché sono dedicate completamente a operazioni di lettura nel volume cluster. Tutte le operazioni di lettura sono gestite dall'istanza primaria. Ogni istanza database di replica di lettura ha il proprio endpoint.

Poiché il volume di archiviazione del cluster è condiviso tra tutte le istanze di un cluster, tutte le istanze di replica di lettura restituiscono gli stessi dati per i risultati delle query con un ritardo di replica minimo. Questo ritardo è in genere molto inferiore a 100 millisecondi dopo che l'istanza primaria scrive un aggiornamento, sebbene possa essere un po' più lungo quando il volume delle operazioni di scrittura è molto elevato.

La disponibilità di una o più istanze di replica di lettura in diverse zone di disponibilità può aumentare la disponibilità, poiché le repliche di lettura fungono da destinazioni di failover per l'istanza primaria.

Pertanto, se si verifica un errore nell'istanza primaria, Neptune promuove un'istanza di replica di lettura a diventare l'istanza primaria. Quando ciò accade, si verifica una breve interruzione mentre l'istanza promossa viene riavviata, durante la quale le richieste di lettura e scrittura effettuate all'istanza primaria hanno esito negativo restituendo un'eccezione.

Al contrario, se il cluster database non include istanze di replica di lettura, il cluster database rimane non disponibile quando l'istanza primaria ha esito negativo finché non viene ricreata. La ricreazione dell'istanza primaria richiede molto più tempo rispetto alla promozione di un'istanza di replica di lettura.

Per garantire un'elevata disponibilità, è consigliabile creare una o più istanze di replica di lettura che abbiano la stessa classe di istanza database dell'istanza primaria e si trovino in zone di disponibilità diverse rispetto all'istanza primaria. Per informazioni, consulta Tolleranza ai guasti di un cluster database Neptune..

Utilizzando la console, puoi creare un'implementazione Multi-AZ semplicemente specificando AZ multiple durante la creazione di un cluster database. Se un cluster database si trova in una sola zona di disponibilità, è possibile renderlo un cluster database Multi-AZ aggiungendo una replica Neptune in una zona di disponibilità diversa.

> **Note**
>
> Non è possibile creare un'istanza di replica di lettura crittografata per un cluster database Neptune non crittografato o un'istanza di replica di lettura non crittografata per un cluster database Neptune crittografato.

Per informazioni dettagliate su come creare un'istanza database di replica di lettura Neptune, consulta Creazione di un'istanza reader Neptune mediante la console.

Dimensionamento delle istanze database in un cluster database Neptune

Dimensionare le istanze del cluster database Neptune in base ai requisiti di CPU e memoria. Il numero di vCPU di un'istanza determina il numero di thread di query che gestiscono le query in entrata. La quantità di memoria di un'istanza determina la dimensione della cache del buffer, utilizzata per archiviare copie delle pagine di dati recuperate dal volume di archiviazione sottostante.

Ogni istanza database Neptune ha un numero di thread di query pari a 2 volte il numero di vCPU dell'istanza. Un'istanza `r5.4xlarge`, ad esempio, con 16 vCPU, ha 32 thread di query e può quindi elaborare 32 query contemporaneamente.

Le query aggiuntive che arrivano mentre tutti i thread di query sono occupati vengono inserite in una coda sul lato server ed elaborate in modo FIFO non appena i thread di query diventano disponibili. Questa coda lato server può contenere circa 8000 richieste in sospeso. Una volta piena, Neptune risponde alle richieste aggiuntive con `ThrottlingException`. Puoi monitorare il numero di richieste in sospeso con la `MainRequestQueuePendingRequests` CloudWatch metrica o utilizzando l'endpoint Gremlin Query Status con il parametro. `includeWaiting`

Il tempo di esecuzione delle query dal punto di vista del client include il tempo trascorso in coda, oltre al tempo impiegato per eseguire effettivamente la query.

Un carico di scrittura simultaneo sostenuto che utilizza tutti i thread di query dell'istanza database primaria indica idealmente un utilizzo della CPU pari o superiore al 90%, il che indica che tutti i thread di query sul server sono attivamente impegnati a svolgere un lavoro utile. Tuttavia, l'utilizzo effettivo della CPU è spesso leggermente inferiore, anche con un carico di scrittura simultaneo sostenuto. Ciò è in genere dovuto al fatto che i thread di query sono in attesa del completamento delle operazioni di I/O sul volume di archiviazione sottostante. Neptune utilizza le scritture quorum che creano sei copie dei dati in tre zone di disponibilità e quattro di questi sei nodi di archiviazione devono riconoscere una scrittura perché sia considerata durevole. Mentre un thread di query attende questo quorum dal volume di archiviazione, viene bloccato, riducendo così l'utilizzo della CPU.

Se si ha un carico di scrittura seriale in cui si esegue una scrittura dopo l'altra e si attende il completamento della prima scrittura prima di iniziare la successiva, è possibile aspettarsi che l'utilizzo della CPU sia ancora inferiore. La quantità esatta dipenderà dal numero di vCPU e thread di query (maggiore è il numero di thread di query, minore sarà la CPU complessiva per query), con una certa riduzione causata dall'attesa dell'I/O.

Per ulteriori informazioni su come dimensionare al meglio le istanze database, consulta Scelta del tipo di istanza database Neptune giusto. Per i prezzi di ogni tipo di istanza, consulta la pagina dei prezzi di Neptune.

Monitoraggio delle prestazioni delle istanze database in Neptune

Puoi utilizzare le CloudWatch metriche di Neptune per monitorare le prestazioni delle tue istanze DB e tenere traccia della latenza delle query osservata dal client. Per informazioni, consulta Utilizzo CloudWatch per monitorare le prestazioni delle istanze DB in Neptune.

Archiviazione, affidabilità e disponibilità di Amazon Neptune

Amazon Neptune utilizza un'architettura di archiviazione distribuita e condivisa che si adatta automaticamente all'aumentare delle esigenze di archiviazione del database.

I dati di Neptune sono archiviati in un volume cluster che è un singolo volume virtuale che utilizza unità SSD Non-Volatile Memory Express (NVMe). Il volume cluster è costituito da una raccolta di blocchi logici noti come segmenti. A ciascuno di questi segmenti vengono assegnati 10 gigabyte (GB) di archiviazione. I dati in ogni segmento vengono replicati in sei copie, che vengono quindi allocate in tre zone di disponibilità (AZ) nella regione AWS in cui risiede il cluster database.

Quando viene creato un cluster database Neptune, gli viene assegnato un singolo segmento di 10 GB. Man mano che il volume di dati aumenta e supera lo spazio di archiviazione attualmente allocato, Neptune espande automaticamente il volume cluster aggiungendo nuovi segmenti. Un volume di cluster Neptune può crescere fino a una dimensione massima di 128 tebibyte (TiB) in tutte le regioni supportate tranne la Cina e GovCloud, dove è limitato a 64 TiB. Per i rilasci del motore precedenti a Rilascio: 1.0.2.2 (09/03/2020), tuttavia, la dimensione dei volumi cluster è limitata a 64 TiB in tutte le regioni.

Il volume cluster database contiene tutti i dati utente, gli indici e i dizionari (descritti nella sezione Modello di dati a grafo di Neptune) oltre ai metadati interni come i log delle transazioni interni. Tutti questi dati del grafo, inclusi gli indici e i log interni, non possono superare la dimensione massima del volume cluster.

Opzione Archiviazione ottimizzata per l'I/O

Neptune offre due modelli di prezzo per l'archiviazione:

- Archiviazione standard: l'archiviazione standard fornisce archiviazione di database conveniente per applicazioni con un numero di operazioni di I/O da moderato a basso.
- Archiviazione ottimizzata per l'I/O: con l'archiviazione ottimizzata per I/O, si paga solo per l'archiviazione utilizzata, a un costo maggiore rispetto all'archiviazione standard e non è previsto alcun pagamento per l'I/O utilizzato.

  L'archiviazione ottimizzata per l'I/O è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O a un costo prevedibile, con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

  Per ulteriori informazioni, consulta Archiviazione ottimizzata per l'I/O.

Allocazione dell'archiviazione di Neptune

Anche se un volume cluster Neptune può aumentare fino a 128 TiB (o 64 TiB in alcune regioni), viene addebitato solo lo spazio effettivamente allocato. Lo spazio totale allocato è determinato dal livello più alto di archiviazione, ovvero la quantità massima allocata al volume cluster in qualsiasi momento durante la sua esistenza.

Ciò significa che anche se i dati utente vengono rimossi da un volume cluster, ad esempio tramite una query di rilascio come `g.V().drop()`, lo spazio totale allocato rimane lo stesso. Neptune ottimizza automaticamente lo spazio allocato inutilizzato per il riutilizzo in futuro.

Oltre ai dati utente, altri due tipi di contenuti occupano spazio di archiviazione interno, vale a dire i dati del dizionario e i log delle transazioni interni. Sebbene i dati del dizionario siano archiviati con i dati del grafo, persistono a tempo indeterminato, anche quando i dati a grafo supportati sono stati eliminati, il che significa che le voci possono essere riutilizzate se i dati vengono reintrodotti. I dati dei log interni vengono archiviati in uno spazio di archiviazione interno separato con un proprio limite massimo. Quando un log interno scade, lo spazio di archiviazione occupato può essere riutilizzato per altri log, ma non per i dati del grafo. La quantità di spazio interno che è stata allocata per i log è inclusa nello spazio totale riportato dalla metrica. `VolumeBytesUsed` CloudWatch

Vedi Best practice per l'archiviazione per trovare il modo di ridurre al minimo lo spazio di archiviazione allocato e di riutilizzare lo spazio.

Fatturazione dell'archiviazione Neptune

I costi di archiviazione vengono fatturati in base al livello più alto di archiviazione, come descritto sopra. Sebbene i dati vengano replicati in sei copie, viene fatturata solo una copia dei dati.

Puoi determinare qual è l'attuale limite massimo di archiviazione del tuo cluster DB monitorando la `VolumeBytesUsed` CloudWatch metrica (vedi). Monitoraggio di Neptune tramite Amazon CloudWatch

Altri fattori che possono influire sui costi di archiviazione di Neptune includono gli snapshot e il backup del database, che vengono fatturati separatamente come archiviazione di backup e si basano sui costi di archiviazione di Neptune (vedi Metriche di CloudWatch utili per la gestione dell'archiviazione di backup di Neptune).

Se si crea un clone del database, tuttavia, il clone punta allo stesso volume cluster utilizzato dal cluster database stesso, quindi non sono previsti costi di archiviazione aggiuntivi per i dati originali.

Le modifiche successive al clone utilizzano il copy-on-write protocollo e comportano costi di storage aggiuntivi.

Per ulteriori informazioni sui prezzi di Neptune, consulta Prezzi di Amazon Neptune.

Best practice per l'archiviazione di Neptune

Poiché alcuni tipi di dati utilizzano un'archiviazione permanente in Neptune, utilizzare queste best practice per evitare grandi picchi di crescita dello spazio di archiviazione:

- Quando si progetta il modello di dati a grafo, evitare il più possibile di utilizzare chiavi di proprietà e valori rivolti all'utente di natura temporanea.
- Se si intende apportare modifiche al modello di dati, non caricare i dati in un cluster database esistente utilizzando il nuovo modello finché non si sono cancellati i dati in quel cluster database utilizzando l'API di reimpostazione rapida. La cosa migliore è spesso caricare dati che utilizzano un nuovo modello in un nuovo cluster database.
- Le transazioni che operano su grandi quantità di dati generano log interni di dimensioni corrispondenti, che possono aumentare in modo permanente il livello più alto dello spazio dei log interni. Ad esempio, una singola transazione che elimina tutti i dati del cluster database potrebbe generare un enorme log interno che richiederebbe l'allocazione di una grande quantità di spazio di archiviazione interno e quindi una riduzione permanente dello spazio disponibile per i dati del grafo.

  Per evitare ciò, suddividere le transazioni di grandi dimensioni in transazioni più piccole e lasciare trascorrere del tempo tra loro in modo che i log interni associati abbiano la possibilità di scadere e liberare lo spazio di archiviazione interno per il riutilizzo nei log successivi.
- Per monitorare la crescita del volume del cluster Neptune, puoi impostare CloudWatch un allarme sulla metrica. `VolumeBytesUsed` CloudWatch Ciò può essere particolarmente utile se i dati raggiungono la dimensione massima del volume cluster. Per ulteriori informazioni, consulta Usare gli CloudWatch allarmi Amazon.

L'unico modo per ridurre lo spazio di archiviazione utilizzato dal cluster database quando si dispone di una grande quantità di spazio allocato inutilizzato è esportare tutti i dati nel grafo e quindi ricaricarli in un nuovo cluster database. Consulta il servizio e l'utilità di esportazione dati di Neptune per esportare facilmente i dati da un cluster database e lo strumento di caricamento in blocco Neptune per reimportare facilmente i dati in Neptune.

> **Note**
>
> La creazione e il ripristino di uno snapshot non riducono la quantità di spazio di archiviazione allocata per il cluster database, poiché uno snapshot conserva l'immagine originale dell'archiviazione sottostante del cluster. Se non viene utilizzata una quantità considerevole dello spazio di archiviazione allocato, l'unico modo per ridurre la quantità di spazio di archiviazione allocata è esportare i dati del grafo e ricaricarli in un nuovo cluster database.

Affidabilità e disponibilità elevata dell'archiviazione Neptune

Amazon Neptune è progettato per garantire affidabilità, durevolezza e tolleranza agli errori.

Il fatto che sei copie dei dati di Neptune siano conservate in tre zone di disponibilità (AZ) garantisce che l'archiviazione dei dati sia altamente durevole, con una probabilità molto bassa di perdita di dati. I dati vengono replicati automaticamente nelle zone di disponibilità indipendentemente dalla presenza di istanze database al loro interno e la quantità di replica è indipendente dal numero di istanze database nel cluster.

Ciò significa che è possibile aggiungere rapidamente una replica di lettura, perché Neptune non crea una nuova copia dei dati del grafo. Invece, la replica di lettura si connette al volume cluster che contiene già i dati. Allo stesso modo, la rimozione di una replica di lettura non rimuove i dati sottostanti.

È possibile eliminare il volume cluster e i relativi dati solo dopo aver eliminato tutte le relative istanze database.

Neptune inoltre rileva automaticamente gli errori nei segmenti che formano il volume cluster. Quando una copia dei dati in un segmento è danneggiata, Neptune ripara immediatamente quel segmento, utilizzando altre copie dei dati all'interno dello stesso segmento per garantire che i dati riparati siano aggiornati. Di conseguenza, Neptune evita la perdita di dati e riduce la necessità di point-in-time eseguire un ripristino per il ripristino in caso di guasto del disco.

Connessione agli endpoint Amazon Neptune

Amazon Neptune utilizza un cluster di istanze database anziché una singola istanza. Ogni connessione Neptune viene gestita da un'istanza database specifica. Quando ti connetti a un cluster Neptune, il nome host e la porta specificati puntano a un handler intermedio chiamato endpoint. Un endpoint è un URL che contiene un indirizzo host e una porta. Gli endpoint Neptune utilizzano connessioni TLS/SSL (Transport Layer Security/Secure Sockets Layer) crittografate.

Neptune utilizza il meccanismo degli endpoint per astrarre queste connessioni in modo da non dover codificare i nomi host o scrivere una propria logica per il reindirizzamento delle connessioni quando alcune istanze database non sono disponibili.

Usando gli endpoint puoi associare ogni connessione all'istanza o al gruppo di istanze appropriato in base al caso d'uso. Gli endpoint personalizzati consentono di connettersi a sottoinsiemi di istanze database. I seguenti endpoint sono disponibili in un cluster .database Neptune.

Endpoint del cluster Neptune

Per endpoint del cluster si intende un endpoint per un cluster database Neptune che si connette all'istanza database primaria corrente di quel cluster. Ciascun cluster database Neptune ha un endpoint del cluster e un'istanza database primaria.

L'endpoint del cluster fornisce un failover del cluster per connessioni di lettura-scrittura al cluster database. Usa l'endpoint del cluster per tutte le operazioni di scrittura sul cluster DB, inclusi aggiornamenti, inserzioni, eliminazioni e modifiche al linguaggio di definizione dati (DDL). Puoi anche utilizzare l'endpoint del cluster per le operazioni di lettura, come ad esempio le query.

In caso di errore dell'istanza database primaria corrente di un cluster database, Neptune esegue automaticamente il failover su una nuova istanza database primaria. Durante un failover, il cluster database continua a servire le richieste di connessione all'endpoint del cluster dalla nuova istanza database primaria, riducendo al minimo l'interruzione del servizio.

L'esempio seguente mostra un endpoint del cluster per un cluster database Neptune.

```
mydbcluster.cluster-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Endpoint di lettura Neptune

Per endpoint di lettura si intende un endpoint per un cluster database Neptune che si connette a una delle repliche Neptune disponibili per quel cluster. Ogni cluster database Neptune ha un endpoint

di lettura. Se sono presenti più repliche Neptune, l'endpoint di lettura instrada ciascuna richiesta di connessione a una delle repliche Neptune.

L'endpoint lettore fornisce l'instradamento round robin per connessioni di sola lettura al cluster database. Puoi utilizzare l'endpoint di lettura per le operazioni di lettura, come ad esempio le query .

Non è possibile utilizzare l'endpoint di lettura per le operazioni di scrittura a meno che non si disponga di un cluster a istanza singola (un cluster senza repliche di lettura). Solo in questo caso, il lettore può essere utilizzato sia per le operazioni di scrittura che per le operazioni di lettura.

L'instradamento round robin dell'endpoint lettore funziona cambiando l'host a cui punta la voce DNS. Ogni volta che si risolve il DNS, ottieni un IP differente e vengono aperte le connessioni verso tali IP. Una volta stabilita una connessione, tutte le richieste per quella connessione vengono inviate allo stesso host. Il client deve creare una nuova connessione e risolvere di nuovo il record DNS per ottenere una connessione a una replica di lettura potenzialmente differente.

> **Note**
>
> WebSockets le connessioni vengono spesso mantenute attive per lunghi periodi. Per ottenere diverse repliche di lettura, procedi nel seguente modo:
>
> - Assicurati che il client risolva la voce DNS ogni volta che si connette.
> - Chiudi la connessione e riconnettiti.

Vari software client potrebbero risolvere il DNS in modi diversi. Ad esempio, se il client risolve il DNS e quindi usa l'IP per ogni connessione, indirizza tutte le richieste a un unico host.

Il caching DNS per client o proxy risolve il nome DNS per lo stesso endpoint dalla cache. Questo è un problema sia per gli scenari di instradamento round robin che di failover.

> **Note**
>
> Disattiva qualunque impostazione di caching DNC per forzare ogni volta la risoluzione DNS.

Il cluster database distribuisce le richieste di connessione all'endpoint di lettura fra le repliche Neptune disponibili. Se il cluster database contiene solo un'istanza database primaria, l'endpoint lettore serve le richieste di connessione dall'istanza database primaria. Se per quel cluster database

viene creata una replica Neptune, l'endpoint di lettura continua a servire le richieste di connessione all'endpoint di lettura dalla nuova replica Neptune, riducendo al minimo l'interruzione del servizio.

L'esempio seguente mostra un endpoint di lettura per un cluster database Neptune.

```
mydbcluster.cluster-ro-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Endpoint dell'istanza Neptune

Per endpoint dell'istanza si intende un endpoint per un'istanza database in un cluster database Neptune che si connette a quella specifica istanza. Ciascuna istanza database di un cluster database, a prescindere dal tipo, ha un proprio endpoint dell'istanza esclusivo. Pertanto, è presente un endpoint dell'istanza per l'istanza database primaria attuale del cluster database. Inoltre, è presente un endpoint dell'istanza per ognuna delle repliche Neptune nel cluster database.

L'endpoint dell'istanza fornisce controllo diretto sulle connessioni al cluster database, per scenari in cui l'utilizzo dell'endpoint del cluster o dell'endpoint di lettura potrebbe non essere appropriato. Ad esempio, l'applicazione client potrebbe richiedere un bilanciamento del carico granulare in base al tipo di carico di lavoro. In questo caso, è possibile configurare più client per connettersi alle repliche Neptune in un cluster database per distribuire i carichi di lavoro in lettura.

L'esempio seguente mostra un endpoint dell'istanza per un'istanza database in un cluster database Neptune.

```
mydbinstance.123456789012.us-east-1.neptune.amazonaws.com:8182
```

Endpoint personalizzati Neptune

L'endpoint personalizzato per un cluster Neptune rappresenta un set di istanze database selezionate. Quando ti connetti all'endpoint, Neptune sceglie una delle istanze del gruppo per gestire la connessione. Puoi definire a quali istanze si riferisce questo endpoint e a quale scopo serve l'endpoint.

Un cluster database Neptune non ha endpoint personalizzati finché non ne crei uno ed è possibile creare fino a cinque endpoint personalizzati per ogni cluster Neptune con provisioning.

L'endpoint personalizzato fornisce le connessioni ai database con bilanciamento del carico sulla base di criteri diversi dalla capacità di sola lettura o di lettura-scrittura delle istanze database. Dal momento che è possibile eseguire la connessione a qualsiasi istanza database associata all'endpoint, assicurarsi che tutte le istanze all'interno del gruppo condividano le stesse caratteristiche

di prestazioni e capacità di memoria. Quando usi gli endpoint personalizzati, in genere non utilizzi l'endpoint di lettura per il cluster.

Questa caratteristica è progettata per gli utenti esperti con tipi specializzati di carichi di lavoro in cui non è pratico mantenere identiche tutte le repliche Neptune nel cluster. Con gli endpoint personalizzati, puoi modificare la capacità delle istanze database utilizzate con ciascuna connessione.

Ad esempio, se si definiscono diversi endpoint personalizzati che si connettono a gruppi di istanze con classi di istanze diverse, è possibile indirizzare gli utenti con esigenze di prestazioni diverse verso gli endpoint più adatti ai loro casi d'uso.

L'esempio seguente mostra un endpoint personalizzato per un'istanza database in un cluster database Neptune.

```
myendpoint.cluster-custom-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Per ulteriori informazioni, consulta Utilizzo degli endpoint personalizzati.

Considerazioni sugli endpoint Neptune

Considera quanto segue quando utilizzi gli endpoint Neptune:

- Prima di utilizzare l'endpoint istanza per la connessione a una specifica istanza database di un cluster database, potresti valutare l'uso dell'endpoint cluster o lettore per il cluster database.

  L'endpoint del cluster e l'endpoint lettore supportano scenari ad alta disponibilità. In caso di errore dell'istanza database primaria di un cluster database, Neptune esegue automaticamente il failover su una nuova istanza database primaria. Questa operazione viene eseguita promuovendo una replica Neptune esistente in una nuova istanza database primaria oppure creando una nuova istanza database primaria. Se si verifica un failover, è possibile utilizzare l'endpoint del cluster per la riconnessione all'istanza database primaria appena creata o promossa oppure ricorrere all'endpoint di lettura per riconnettersi a una delle altre repliche Neptune nel cluster database.

  Anche se scegli di adottare un approccio diverso, puoi comunque assicurarti di effettuare la connessione all'istanza database corretta del cluster database per l'operazione desiderata. A tale scopo, puoi scoprire in modo manuale o programmatico la serie risultante di istanze database nel cluster DB e confermare i tipi di istanze dopo il failover, prima di utilizzare l'endpoint di un'istanza database specifica.

  Per ulteriori informazioni sui failover, consulta Tolleranza ai guasti di un cluster database Neptune..

- L'endpoint di lettura indirizza le connessioni solo a repliche Neptune disponibili in un cluster database Neptune. Non indirizza query specifiche.

  > **Important**
  >
  > Neptune non bilancia il carico.

  Se desideri bilanciare il carico di query per distribuire il carico di lavoro in lettura per un cluster database, devi gestire l'operazione nella tua applicazione. Devi utilizzare endpoint di istanza per connetterti direttamente alle repliche Neptune per bilanciare il carico.

- L'instradamento round robin dell'endpoint lettore funziona cambiando l'host a cui punta la voce DNS. Il client deve creare una nuova connessione e risolvere di nuovo il record DNS per ottenere una connessione a una replica di lettura potenzialmente nuova.

- Durante un failover, l'endpoint di lettura potrebbe indirizzare le connessioni alla nuova istanza database primaria di un cluster database per un breve periodo di tempo, quando una replica Neptune viene promossa a nuova istanza database primaria.

Utilizzo degli endpoint personalizzati in Neptune

Quando aggiungi un'istanza database a un endpoint personalizzato o la si rimuove da un endpoint personalizzato, tutte le connessioni esistenti all'istanza database rimangono attive.

È possibile definire un elenco di istanze database da includere in un endpoint personalizzato (elenco statico) o da escludere dall'endpoint personalizzato (elenco di esclusione). È possibile utilizzare il meccanismo di inclusione/esclusione per suddividere le istanze database in gruppi e assicurarsi che gli endpoint personalizzati coprano tutte le istanze database del cluster. Ogni endpoint personalizzato può contenere solo uno di questi tipi di elenchi.

In AWS Management Console, la scelta è rappresentata dalla casella di controllo Allega le future istanze aggiunte a questo cluster. Quando lasci deselezionata la casella di controllo, l'endpoint personalizzato utilizza un elenco statico contenente solo le istanze database specificate nella finestra

di dialogo. Quando selezioni la casella di controllo, l'endpoint personalizzato utilizza un elenco di esclusione. In questo caso, l'endpoint personalizzato rappresenta tutte le istanze database nel cluster (incluse quelle che aggiungi in futuro) tranne quelle lasciate deselezionate nella finestra di dialogo.

Neptune non modifica le istanze database specificate in questi elenchi statici o di esclusione quando le istanze database modificano i ruoli tra l'istanza primaria e la replica Neptune a causa di un failover o una promozione.

Puoi associare un'istanza database a più di un endpoint personalizzato. Ad esempio, supponiamo di aggiungere una nuova istanza database a un cluster. In questo caso, l'istanza database viene aggiunta a tutti gli endpoint personalizzati per i quali è idonea. L'elenco statico o di esclusione definito determina quale istanza database può essere aggiunta.

Se un endpoint include un elenco statico di istanze database, le repliche Neptune appena aggiunte non vengono inserite nell'endpoint. Al contrario, se l'endpoint ha un elenco di esclusione, le repliche Neptune appena aggiunte vengono inserite nell'endpoint a condizione che non siano indicate nell'elenco di esclusione.

Se una replica Neptune diventa non disponibile, rimane associata ai relativi endpoint personalizzati. Ciò vale sia se la replica sia non integra, arrestata, in fase di riavvio o non disponibile per un altro motivo. Tuttavia, finché non è disponibile, non è possibile connettersi ad essa tramite alcun endpoint.

Dal momento che i cluster Neptune appena creati non hanno endpoint personalizzati, è necessario crearli e gestirli autonomamente. Questo vale anche per i cluster Neptune ripristinati dagli snapshot, poiché gli endpoint personalizzati non sono inclusi nello snapshot. È necessario crearli nuovamente dopo il ripristino e scegliere nuovi nomi di endpoint se il cluster ripristinato si trova nella stessa regione di quello originale.

Creazione di un endpoint personalizzato

Gestire gli endpoint personalizzati utilizzando la console Neptune. A tale scopo, accedere alla pagina dei dettagli del cluster Neptune e utilizzare i controlli nella sezione Endpoint personalizzati.

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Accedere alla pagina dei dettagli del cluster.
3. Scegliere l'azione `Create custom endpoint` nella sezione Endpoint.
4. Scegliere un nome per l'endpoint personalizzato che sia univoco per l'ID utente e la regione. Il nome deve avere una lunghezza massima di 63 caratteri e avere il seguente formato:

endpointName.cluster-custom-*customerDnsIdentifier*.*dnsSuffix*

Poiché i nomi degli endpoint personalizzati non includono il nome del cluster, non è necessario modificarli se si rinomina un cluster. Tuttavia, non è possibile riutilizzare lo stesso nome di endpoint personalizzato per più di un cluster nella stessa regione. Assegna a ciascun endpoint personalizzato un nome univoco tra i cluster di proprietà dell'ID utente all'interno di una determinata regione.

5. Per selezionare un elenco di istanze database che rimane invariato anche quando il cluster si espande, lascia deselezionata la casella di controllo Attach future instances added to this cluster (Collega le istanze che in futuro saranno aggiunte a questo cluster). Quando questa casella di controllo è selezionata, l'endpoint personalizzato aggiunge dinamicamente tutte le nuove istanze che vengono aggiunte al cluster.

Visualizzazione degli endpoint personalizzati

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Accedere alla pagina dei dettagli del cluster database.
3. La sezione Endpoint contiene solo informazioni sugli endpoint personalizzati (i dettagli sugli endpoint integrati sono elencati nella sezione Dettagli principale). Per vedere i dettagli di uno specifico endpoint personalizzato, selezionane il nome per visualizzare la relativa pagina dei dettagli.

Modifica di un endpoint personalizzato

Puoi modificare le proprietà di un endpoint personalizzato per cambiare le istanze database ad esso associate. Puoi anche passare da un elenco statico a un elenco di esclusione.

Non puoi connetterti o utilizzare un endpoint personalizzato mentre sono in corso le modifiche di un'operazione di modifica. Potrebbero essere necessari alcuni minuti dopo la modifica prima che lo stato dell'endpoint torni Disponibile permettendoti di riconnetterti.

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Accedere alla pagina dei dettagli del cluster.
3. Nella sezione Endpoint, scegliere il nome dell'endpoint personalizzato da modificare.

4. Nella pagina dei dettagli dell'endpoint, scegliere l'azione Modifica.

Eliminazione di un endpoint personalizzato

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Accedere alla pagina dei dettagli del cluster.
3. Nella sezione Endpoint, scegliere il nome dell'endpoint personalizzato da eliminare.
4. Nella pagina dei dettagli dell'endpoint, scegliere l'azione Elimina.

Inserimento di un ID personalizzato in una query Neptune Gremlin o SPARQL

Per impostazione predefinita, Neptune assegna un valore `queryId` univoco a ogni query. Puoi utilizzare questo ID per ottenere informazioni su una query in esecuzione (consulta API di stato delle query Gremlin o API di stato delle query SPARQL) o annullarla (consulta Annullamento delle query Gremlin o Annullamento della query SPARQL).

Neptune consente inoltre di specificare il proprio valore `queryId` per una query Gremlin o SPARQL, nell'intestazione HTTP o per una query SPARQL utilizzando l'hint di query `queryId`. L'assegnazione del proprio `queryID` consente di tenere traccia di una query in modo da ottenere lo stato o annullarla.

> **Note**
>
> Questa funzionalità è disponibile a partire dal Rilascio 1.0.1.0.200463.0 (15/10/2019).

Inserimento di un valore **`queryId`** personalizzato tramite l'intestazione HTTP

Per Gremlin e SPARQL, l'intestazione HTTP può essere utilizzata per inserire il proprio valore `queryId` in una query.

Esempio Gremlin

```
curl -XPOST https://your-neptune-endpoint:port \
    -d "{\"gremlin\": \
        \"g.V().limit(1).count()\" , \
        \"queryId\":\"4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47\"  }"
```

Esempio SPARQL

```
curl https://your-neptune-endpoint:port/sparql \
    -d "query=SELECT * WHERE { ?s ?p ?o } " \
       --data-urlencode \
       "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Inserimento di un valore **queryId** personalizzato mediante un hint di query SPARQL

Di seguito è riportato un esempio di come utilizzare l'hint di query `queryId` SPARQL per inserire un valore `queryId` personalizzato in una query SPARQL:

```
curl https://your-neptune-endpoint:port/sparql \
    -d "PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#> \
       SELECT * WHERE { hint:Query hint:queryId \"4d5c4fae-
aa30-41cf-9e1f-91e6b7dd6f47\" \
       {?s ?p ?o}}"
```

Utilizzo del valore **queryId** per controllare lo stato della query

Esempio Gremlin

```
curl https://your-neptune-endpoint:port/gremlin/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Esempio SPARQL

```
curl https://your-neptune-endpoint:port/sparql/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Modalità di laboratorio Neptune

Puoi utilizzare la modalità di laboratorio di Amazon Neptune per abilitare nuove funzionalità presenti nell'attuale rilascio del motore Neptune, ma che non sono ancora pronte per l'uso in produzione e non sono abilitate per impostazione predefinita. In questo modo puoi provare queste caratteristiche negli ambienti di sviluppo e test.

> **Note**
>
> Questa funzionalità è disponibile a partire dal Rilascio 1.0.1.0.200463.0 (15/10/2019).

Utilizzo della modalità di laboratorio Neptune

Utilizza il parametro del cluster database `neptune_lab_mode` per abilitare o disabilitare le funzionalità. A tale scopo, includi *`(feature name)`*`=enabled` o *`(feature name)`*`=disabled` nel valore del parametro `neptune_lab_mode` nel gruppo di parametri del cluster database.

Ad esempio, in questa versione del motore potrebbe essere necessario impostare il parametro `neptune_lab_mode` su `Streams=disabled, ReadWriteConflictDetection=enabled`.

Per informazioni su come modificare il gruppo di parametri del cluster database per il database, consulta Modifica di un gruppo di parametri. Non è possibile modificare il gruppo di parametri del cluster database predefinito; se si sta utilizzando il gruppo predefinito, è necessario creare un nuovo gruppo di parametri del cluster database prima di poter impostare il parametro `neptune_lab_mode`.

> **Note**
>
> Quando apporti una modifica a un parametro statico del cluster database, ad esempio `neptune_lab_mode`, devi riavviare l'istanza primaria (scrittura) del cluster affinché la modifica abbia effetto. Prima del Rilascio: 1.2.0.0 (21/07/2022), tutte le repliche di lettura in un cluster database venivano riavviate automaticamente al riavvio dell'istanza primaria.
> A partire dal Rilascio: 1.2.0.0 (21/07/2022), il riavvio dell'istanza primaria non causa il riavvio delle repliche. Ciò significa che è necessario riavviare ogni istanza separatamente per rilevare una modifica del parametro del cluster database (vedi Gruppi di parametri).

Important

Al momento, se fornisci parametri errati in modalità di laboratorio o se la richiesta non riesce per un altro motivo, potresti non ricevere alcuna notifica dell'errore. È sempre necessario verificare che una richiesta di modifica in modalità di laboratorio abbia avuto esito positivo chiamando l'API di stato come illustrato di seguito:

```
curl -G https://your-neptune-endpoint:port/status
```

I risultati dello stato includono informazioni sulla modalità di laboratorio che mostreranno se le modifiche richieste sono state apportate o meno:

```
{
  "status":"healthy",
  "startTime":"Wed Dec 29 02:29:24 UTC 2021",
  "dbEngineVersion":"development",
  "role":"writer",
  "dfeQueryEngine":"viaQueryHint",
  "gremlin":{"version":"tinkerpop-3.5.2"},
  "sparql":{"version":"sparql-1.1"},
  "opencypher":{"version":"Neptune-9.0.20190305-1.0"},
  "labMode":{
    "ObjectIndex":"disabled",
    "ReadWriteConflictDetection":"enabled"
  },
  "features":{
    "LookupCache":{"status":"Available"},
    "ResultCache":{"status":"disabled"},
    "IAMAuthentication":"disabled",
    "Streams":"disabled",
    "AuditLog":"disabled"
  },
  "settings":{"clusterQueryTimeoutInMs":"120000"}
}
```

Le seguenti funzionalità sono attualmente accessibili in modalità di laboratorio:

Indice OSGP

Neptune può ora mantenere un quarto indice, ovvero l'indice OSGP, che è utile per i set di dati con un gran numero di predicati (consulta Abilitazione di un indice OSGP).

> **Note**
>
> Questa funzionalità è disponibile a partire dal rilascio 1.0.2.1 del motore Neptune.

È possibile abilitare un indice OSGP in un nuovo cluster database Neptune vuoto impostando `ObjectIndex=enabled` nel parametro `neptune_lab_mode` del cluster database. Un indice OSGP può essere abilitato solo in un nuovo cluster database vuoto.

Per impostazione predefinita, l'indice OSPG è disabilitato.

> **Note**
>
> Dopo aver impostato il parametro `neptune_lab_mode` del cluster database in modo da abilitare l'indice OSGP, è necessario riavviare l'istanza di scrittura del cluster affinché la modifica abbia effetto.

> **Warning**
>
> Se si disabilita un indice OSGP abilitato impostando `ObjectIndex=disabled` e successivamente lo si riabilita dopo aver aggiunto altri dati, l'indice non verrà compilato correttamente. La ricompilazione su richiesta dell'indice non è supportata, quindi è necessario abilitare l'indice OSGP solo quando il database è vuoto.

Semantica formalizzata delle transazioni

Neptune ha aggiornato la semantica formale per transazioni simultanee (consulta Semantica delle transazioni in Neptune).

Utilizzare `ReadWriteConflictDetection` come nome nel parametro `neptune_lab_mode` che abilita o disabilita la semantica formalizzata delle transazioni.

Per impostazione predefinita, la semantica formalizzata delle transazioni è già abilitata. Se desideri ripristinare il comportamento precedente, includi `ReadWriteConflictDetection=disabled` nel valore impostato per il parametro `neptune_lab_mode` del cluster database.

Il motore di query alternativo (DFE) di Amazon Neptune

Amazon Neptune dispone di un motore di query alternativo noto come DFE che utilizza risorse di istanze database come core CPU, memoria e I/O in modo più efficiente rispetto al motore Neptune originale.

> **Note**
>
> Con set di dati di grandi dimensioni, il motore DFE potrebbe non funzionare bene con le istanze t3.

Il motore DFE esegue query SPARQL, Gremlin e openCypher e supporta un'ampia varietà di tipi di piani, compresi quelli left-deep, bushy e ibridi. Gli operatori del piano possono richiamare sia operazioni di calcolo, che vengono eseguite su un set riservato di core di calcolo, sia operazioni di I/O, ognuna delle quali viene eseguita sul proprio thread in un pool di thread di I/O.

Il motore DFE utilizza statistiche pregenerate sui dati del grafo Neptune per prendere decisioni informate su come strutturare le query. Per informazioni su come vengono generate queste statistiche, consulta Statistiche DFE.

La scelta del tipo di piano e del numero di thread di calcolo utilizzati viene effettuata automaticamente in base alle statistiche pregenerate e alle risorse disponibili nel nodo head Neptune. L'ordine dei risultati non è predeterminato per i piani con parallelismo di calcolo interno.

Controllo di dove viene utilizzato il motore DFE Neptune

Per impostazione predefinita, il parametro neptune_dfe_query_engine di un'istanza è impostato su `viaQueryHint`, il che fa sì che il motore DFE venga utilizzato solo per le query openCypher e per le query Gremlin e SPARQL che includono esplicitamente l'hint di query `useDFE` impostato su `true`.

È possibile abilitare completamente il motore DFE in modo che venga utilizzato laddove possibile impostando il parametro di istanza `neptune_dfe_query_engine` su `enabled`.

È inoltre possibile disabilitare il motore DFE includendo l'hint di query `useDFE` per una particolare query Gremlin o query SPARQL. Questo hint di query consente di impedire al motore DFE di eseguire quella particolare query.

È possibile determinare se il motore DFE è abilitato o meno in un'istanza utilizzando una chiamata Stato dell'istanza, in questo modo:

```
curl -G https://your-neptune-endpoint:port/status
```

La risposta dello stato specifica quindi se il motore DFE è abilitato o meno:

```
{
  "status":"healthy",
  "startTime":"Wed Dec 29 02:29:24 UTC 2021",
  "dbEngineVersion":"development",
  "role":"writer",
  "dfeQueryEngine":"viaQueryHint",
  "gremlin":{"version":"tinkerpop-3.5.2"},
  "sparql":{"version":"sparql-1.1"},
  "opencypher":{"version":"Neptune-9.0.20190305-1.0"},
  "labMode":{
    "ObjectIndex":"disabled",
    "ReadWriteConflictDetection":"enabled"
  },
  "features":{
    "ResultCache":{"status":"disabled"},
    "IAMAuthentication":"disabled",
    "Streams":"disabled",
    "AuditLog":"disabled"
  },
  "settings":{"clusterQueryTimeoutInMs":"120000"}
}
```

I risultati `explain` e `profile` di Gremlin indicano se una query viene eseguita dal motore DFE. Vedi Informazioni contenute in un report Gremlin `explain` per `explain` e Report di `profile` con DFE per `profile`.

Allo stesso modo, SPARQL `explain` indica se una query SPARQL viene eseguita dal motore DFE. Per ulteriori dettagli, consulta Esempio di output di SPARQL `explain` quando è abilitato il motore DFE e Operatore `DFENode`.

Costrutti di query supportati dal motore DFE Neptune

Attualmente, il motore DFE Neptune supporta un sottoinsieme di costrutti di query SPARQL e Gremlin.

Per SPARQL, si tratta del sottoinsieme di modelli del grafo di base congiuntivi.

Per Gremlin, si tratta generalmente del sottoinsieme di query che contengono una catena di attraversamenti che non contengono alcuni dei passaggi più complessi.

Per vedere se una query viene eseguita per intero o in parte dal motore DFE, procedere nel modo seguente:

- In Gremlin, i risultati `explain` e `profile` indicano quali parti di una query vengono eseguite dal motore DFE, se presenti. Vedi Informazioni contenute in un report Gremlin `explain` per `explain` e Report di `profile` con DFE per `profile`. Consulta anche Ottimizzazione delle query Gremlin con `explain` e `profile`.

  I dettagli sul supporto del motore Neptune per i singoli passaggi di Gremlin sono documentati in Supporto dei passaggi Gremlin.
- Allo stesso modo, SPARQL `explain` indica se una query SPARQL viene eseguita dal motore DFE. Per ulteriori dettagli, consulta Esempio di output di SPARQL `explain` quando è abilitato il motore DFE e Operatore `DFENode`.

Gestione delle statistiche utilizzabili dal motore DFE Neptune

> **Note**
>
> Il supporto per openCypher dipende dal motore di query DFE di Neptune.
> Il motore DFE è stato disponibile per la prima volta in modalità di laboratorio nel rilascio 1.0.3.0 del motore Neptune e, a partire dal rilascio 1.0.5.0 del motore Neptune, è stato abilitato per impostazione predefinita, ma solo per l'uso con hint di query e per il supporto di openCypher.
> A partire dal rilascio 1.1.1.0 del motore Neptune, il motore DFE non è più in modalità Lab e ora viene controllato con il parametro di istanza neptune_dfe_query_engine nel gruppo di parametri database di un'istanza.

Il motore DFE utilizza le informazioni sui dati del grafo Neptune per trovare compromessi efficaci al momento di pianificare l'esecuzione delle query. Queste informazioni assumono la forma di statistiche che includono i cosiddetti insiemi di caratteristiche, nonché statistiche sui predicati utili per la pianificazione delle query.

A partire dalla versione 1.2.1.0 del motore, puoi recuperare informazioni di riepilogo sul tuo grafico da queste statistiche utilizzando l'API o l'GetGraphSummaryendpoint. `summary`

Queste statistiche DFE vengono attualmente rigenerate ogni volta che viene modificato più del 10% dei dati nel grafo o quando le ultime statistiche risalgono a più di 10 giorni fa. Tuttavia, questi trigger potrebbero cambiare in futuro.

> **Note**
>
> La generazione di statistiche è disabilitata nelle istanze `T3` e `T4g` in quanto può superare la capacità di memoria di tali tipi di istanze.

È possibile gestire la generazione di statistiche DFE tramite uno dei seguenti endpoint:

- `https://`*`your-neptune-host`*`:`*`port`*`/rdf/statistics` (per SPARQL).
- `https://`*`your-neptune-host`*`:`*`port`*`/propertygraph/statistics` (per Gremlin e openCypher) e la relativa versione alternativa: `https://`*`your-neptune-host`*`:`*`port`*`/pg/statistics`.

> **Note**
>
> A partire dal rilascio 1.1.1.0 del motore, l'endpoint delle statistiche Gremlin (`https://your-neptune-host:port/gremlin/statistics`) è stato deprecato a favore dell'endpoint `propertygraph` o `pg`. È ancora supportato per la compatibilità con le versioni precedenti, ma potrebbe essere rimosso nei rilasci futuri.
> A partire dal rilascio 1.2.1.0 del motore, l'endpoint delle statistiche SPARQL (`https://your-neptune-host:port/sparql/statistics`) è stato deprecato a favore dell'endpoint `rdf`. È ancora supportato per la compatibilità con le versioni precedenti, ma potrebbe essere rimosso nei rilasci futuri.

Negli esempi seguenti, `$STATISTICS_ENDPOINT` indica uno qualsiasi di questi URL di endpoint.

> **Note**
>
> Se un endpoint delle statistiche DFE si trova in un'istanza di lettura, le uniche richieste che può elaborare sono le richieste di stato. Le altre richieste avranno esito negativo con `ReadOnlyViolationException`.

Limiti di dimensione per la generazione di statistiche DFE

Attualmente, la generazione di statistiche DFE si interrompe se viene raggiunto uno dei seguenti limiti di dimensione:

- Il numero di set di caratteristiche generati non può superare 50.000.
- Il numero di statistiche sui predicati generate non può superare un milione.

Questi limiti possono cambiare.

Stato corrente delle statistiche DFE

È possibile controllare lo stato corrente delle statistiche DFE utilizzando la seguente richiesta `curl`:

```
curl -G "$STATISTICS_ENDPOINT"
```

La risposta a una richiesta di status include i seguenti campi:

- `status`: codice HTTP restituito della richiesta. Se la richiesta è riuscita, il codice è `200`. Per visualizzare un elenco di errori comuni, consulta Errori comuni.
- `payload`:
  - `autoCompute`: (booleano) indica se la generazione automatica di statistiche è abilitata o meno.
  - `active`: (booleano) indica se la generazione di statistiche DFE è abilitata o meno.
  - `statisticsId` : riporta l'ID dell'esecuzione corrente della generazione delle statistiche. Il valore `-1` indica che non sono state generate statistiche.
  - `date`: l'ora UTC in cui sono state generate le statistiche DFE più recentemente, nel formato ISO 8601.

    > **Note**
    >
    > Prima del rilascio 1.2.1.0 del motore, questo valore era rappresentato con precisione al minuto, ma dal rilascio 1.2.1.0 in poi, viene rappresentato con precisione al millisecondo (ad esempio, `2023-01-24T00:47:43.319Z`).

  - `note`: una nota sui problemi nel caso in cui le statistiche non siano valide.
  - `signatureInfo`: contiene informazioni sui set di caratteristiche generati nelle statistiche (prima del rilascio 1.2.1.0 del motore, questo campo era denominato `summary`). In genere non sono direttamente utilizzabili:
    - `signatureCount`: numero totale di firme in tutti i set di caratteristiche.
    - `instanceCount`: numero totale di istanze di set di caratteristiche.
    - `predicateCount`: numero totale di predicati univoci.

La risposta a una richiesta di stato quando non sono state generate statistiche è simile alla seguente:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : -1
   }
}
```

Se le statistiche DFE sono disponibili, la risposta è simile alla seguente:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : true,
    "statisticsId" : 1588893232718,
    "date" : "2020-05-07T23:13Z",
    "summary" : {
      "signatureCount" : 5,
      "instanceCount" : 1000,
      "predicateCount" : 20
    }
  }
}
```

Se la generazione delle statistiche DFE non è riuscita, ad esempio perché ha superato il limite di dimensione delle statistiche, la risposta è simile alla seguente:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : 1588713528304,
    "date" : "2020-05-05T21:18Z",
    "note" : "Limit reached: Statistics are not available"
  }
}
```

Disabilitazione della generazione automatica di statistiche DFE

Per impostazione predefinita, la generazione automatica delle statistiche DFE viene abilitata quando si abilita DFE.

È possibile disabilitare la generazione automatica come segue:

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "disableAutoCompute" }'
```

Se la richiesta ha esito positivo, il codice di risposta HTTP è `200` e la risposta è:

```
{
  "status" : "200 OK"
}
```

È possibile verificare che la generazione automatica è disabilitata inviando una richiesta di stato e verificando che il campo `autoCompute` nella risposta sia impostato su `false`.

La disabilitazione della generazione automatica delle statistiche non interrompe il calcolo delle statistiche in corso.

Se si effettua una richiesta per disabilitare la generazione automatica in un'istanza di lettura anziché nell'istanza di scrittura del cluster database, la richiesta ha esito negativo con il codice HTTP restituito 400 e un output simile al seguente:

```
{
  "detailedMessage" : "Writes are not permitted on a read replica instance",
  "code" : "ReadOnlyViolationException",
  "requestId":"8eb8d3e5-0996-4a1b-616a-74e0ec32d5f7"
}
```

Per visualizzare un elenco di altri errori comuni, consulta Errori comuni.

Riabilitazione della generazione automatica di statistiche DFE

Per impostazione predefinita, la generazione automatica delle statistiche DFE è già abilitata quando si abilita DFE. Se si disabilita la generazione automatica, è possibile riabilitarla in un secondo momento come segue:

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "enableAutoCompute" }'
```

Se la richiesta ha esito positivo, il codice di risposta HTTP è `200` e la risposta è:

```
{
  "status" : "200 OK"
}
```

È possibile verificare che la generazione automatica è abilitata inviando una richiesta di stato e verificando che il campo `autoCompute` nella risposta sia impostato su `true`.

Attivazione manuale della generazione di statistiche DFE

È possibile avviare manualmente la generazione di statistiche DFE nel modo seguente:

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "refresh" }'
```

Se la richiesta ha esito positivo, l'output è il seguente, con il codice HTTP restituito 200:

```
{
  "status" : "200 OK",
  "payload" : {
    "statisticsId" : 1588893232718
  }
}
```

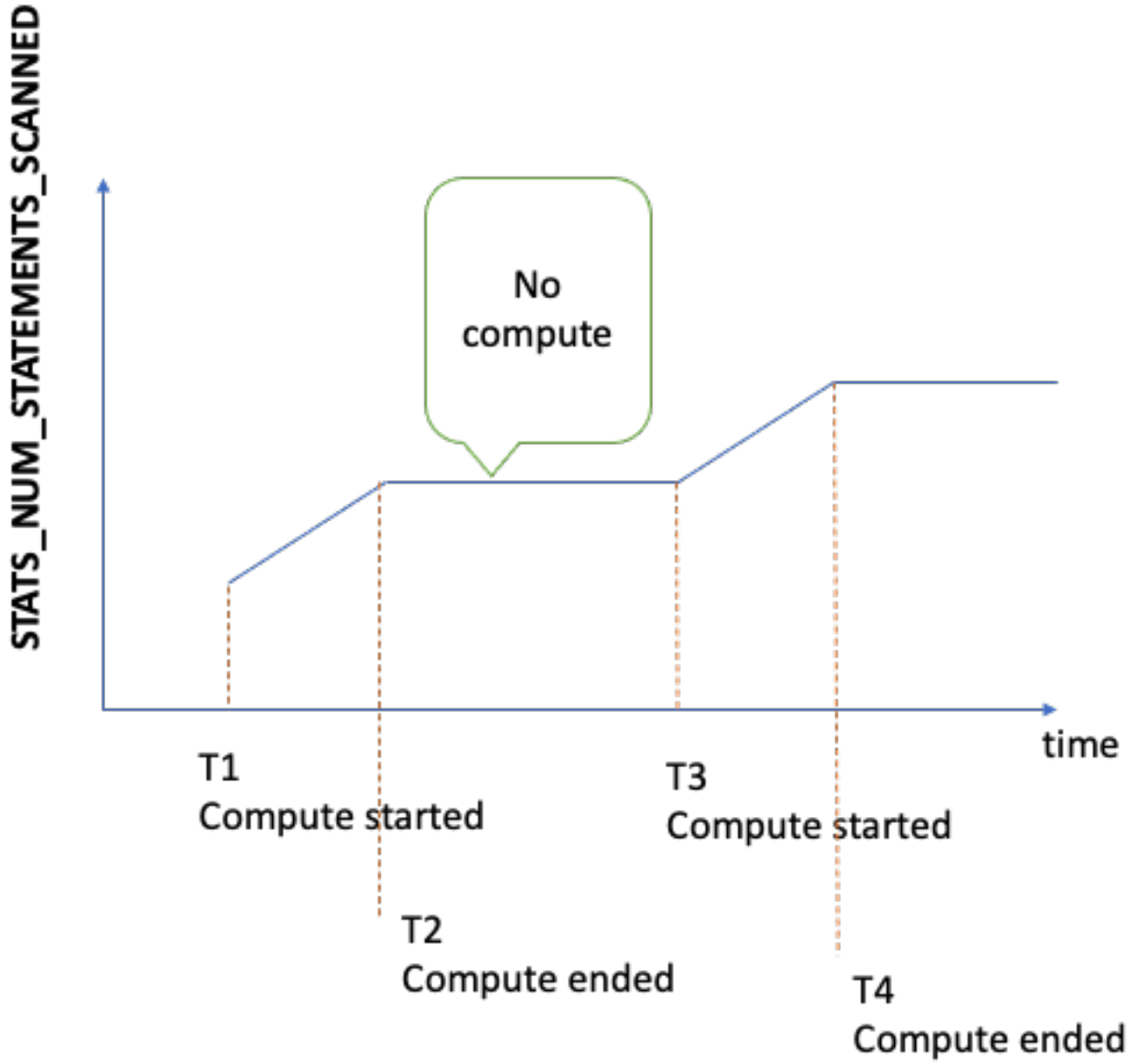
Il valore `statisticsId` nell'output è l'ID dell'esecuzione della generazione di statistiche attualmente in corso. Se un'esecuzione era già in corso al momento della richiesta, la richiesta restituisce l'ID di tale esecuzione anziché avviarne una nuova. È possibile eseguire una sola generazione di statistiche alla volta.

Se si verifica un failover durante la generazione delle statistiche DFE, il nuovo nodo di scrittura rileverà l'ultimo checkpoint elaborato e riprenderà l'esecuzione delle statistiche da lì.

Utilizzo della **`StatsNumStatementsScanned`** CloudWatch metrica per monitorare il calcolo delle statistiche

La `StatsNumStatementsScanned` CloudWatch metrica restituisce il numero totale di istruzioni analizzate per il calcolo delle statistiche dall'avvio del server. Viene aggiornata ad ogni sezione di calcolo delle statistiche.

Ogni volta che viene attivato il calcolo delle statistiche, questo numero aumenta e quando non viene eseguito alcun calcolo, rimane costante. L'esame di un grafico dei valori di `StatsNumStatementsScanned` nel tempo offre quindi un quadro abbastanza chiaro di quando è stato effettuato il calcolo delle statistiche e con quale velocità:



Durante il calcolo, la pendenza del grafico mostra la velocità (maggiore è la pendenza, più velocemente vengono calcolate le statistiche).

Se il grafico è semplicemente una linea piatta a 0, la funzionalità delle statistiche è stata abilitata, ma non è stata calcolata alcuna statistica. Se la funzionalità delle statistiche è stata disabilitata o se si utilizza una versione del motore che non supporta il calcolo delle statistiche, `StatsNumStatementsScanned` non esiste.

Come accennato in precedenza, è possibile disabilitare il calcolo delle statistiche utilizzando l'API delle statistiche, ma lasciarlo disabilitato può far sì che le statistiche non vengano aggiornate, il che a sua volta può comportare una generazione del piano di query inappropriata per il motore DFE.

Monitoraggio di Neptune tramite Amazon CloudWatchPer informazioni sull'uso CloudWatch, vedere.

Utilizzo dell'autenticazione AWS Identity and Access Management (IAM) con gli endpoint statistici DFE

È possibile accedere agli endpoint delle statistiche DFE in modo sicuro con l'autenticazione IAM utilizzando awscurl o qualsiasi altro strumento che funzioni con HTTPS e IAM. Per vedere come configurare le credenziali corrette, consulta Utilizzo di `awscurl` con credenziali temporanee per connettersi in modo sicuro a un cluster database con autenticazione IAM abilitata. Dopo aver eseguito questa operazione, è possibile effettuare una richiesta di stato come questa:

```
awscurl "$STATISTICS_ENDPOINT" \
    --region (your region) \
    --service neptune-db
```

Oppure, ad esempio, è possibile creare un file JSON denominato `request.json` che contiene:

```
{ "mode" : "refresh" }
```

È quindi possibile avviare manualmente la generazione di statistiche nel modo seguente:

```
awscurl "$STATISTICS_ENDPOINT" \
    --region (your region) \
    --service neptune-db \
    -X POST -d @request.json
```

Eliminazione delle statistiche DFE

È possibile eliminare tutte le statistiche nel database effettuando una richiesta HTTP DELETE all'endpoint delle statistiche:

```
curl -X "DELETE" "$STATISTICS_ENDPOINT"
```

I codici HTTP restituiti validi sono:

- `200`: l'eliminazione è riuscita.

In questo caso, una risposta tipica è la seguente:

```
{
  "status" : "200 OK",
  "payload" : {
      "active" : false,
      "statisticsId" : -1
  }
}
```

- `204`: non c'erano statistiche da eliminare.

  In questo caso, la risposta è vuota (nessuna risposta).

Se si invia una richiesta di eliminazione a un endpoint di statistiche su un nodo di lettura, viene generata un'eccezione `ReadOnlyViolationException`.

Codici di errore comuni per la richiesta di statistiche DFE

Di seguito è riportato un elenco di errori comuni che possono verificarsi quando si effettua una richiesta a un endpoint di statistiche:

- `AccessDeniedException`: codice restituito: `400`. Messaggio `Missing Authentication Token`.
- `BadRequestException` (per Gremlin e openCypher): codice restituito: `400`. Messaggio `Bad route: /pg/statistics`.
- `BadRequestException` (per dati RDF): codice restituito: `400`. Messaggio `Bad route: /rdf/statistics`.
- `InvalidParameterException`: codice restituito: `400`. Messaggio `Statistics command parameter 'mode' has unsupported value '`*`the invalid value`*`'`.
- `MissingParameterException`: codice restituito: `400`. Messaggio `Content-type header not specified.`.
- `ReadOnlyViolationException`: codice restituito: `400`. Messaggio `Writes are not permitted on a read replica instance`.

Ad esempio, se si effettua una richiesta quando il motore DFE e le statistiche non sono abilitati, si otterrà una risposta simile alla seguente:

```
{
  "code" : "BadRequestException",
  "requestId" : "b2b8f8ee-18f1-e164-49ea-836381a3e174",
  "detailedMessage" : "Bad route: /sparql/statistics"
}
```

Creazione di un breve report di riepilogo sul grafo

L'API di riepilogo del grafo di Neptune recupera le seguenti informazioni sul grafo:

- Per i grafi di proprietà (PG), l'API di riepilogo del grafo restituisce un elenco di sola lettura di etichette di nodi ed archi, nonché di chiavi di proprietà, insieme al conteggio di nodi, archi e proprietà.
- Per i grafi RDF (Resource Description Framework), l'API di riepilogo del grafo restituisce un elenco di sola lettura di classi e chiavi di predicato, insieme al conteggio di quadruple, soggetti e predicati.

> **Note**
>
> L'API di riepilogo del grafo è stata introdotta nel rilascio 1.2.1.0 del motore Neptune.

Con l'API di riepilogo del grafo, è possibile acquisire rapidamente una comprensione di alto livello della dimensione e del contenuto dei dati del grafo. È inoltre possibile utilizzare l'API in modo interattivo all'interno di un notebook Neptune utilizzando il comando magic `%summary` di Neptune Workbench. In un'applicazione a grafo, l'API può essere utilizzata per migliorare i risultati della ricerca fornendo le etichette dei nodi o degli archi individuati come parte della ricerca.

I dati di riepilogo del grafo sono ricavati dalle statistiche DFE calcolate dal motore DFE Neptune durante il runtime e sono disponibili ogni volta che sono disponibili le statistiche DFE. Le statistiche sono abilitate per impostazione predefinita quando si crea un nuovo cluster database Neptune.

> **Note**
>
> La generazione di statistiche è disabilitata sui tipi di istanza `t3` e `t4` (ovvero sui tipi di istanza `db.t3.medium` e `db.t4g.medium`) per risparmiare memoria. Di conseguenza, i dati di riepilogo del grafo non sono disponibili nemmeno su questi tipi di istanza.

È possibile controllare lo stato delle statistiche DFE utilizzando l'API di stato delle statistiche. Finché la generazione automatica delle statistiche non viene disabilitata, le statistiche vengono aggiornate automaticamente periodicamente.

Se si vuole essere sicuri che le statistiche siano il più aggiornate possibile quando si richiede un riepilogo del grafo, è possibile attivare manualmente un aggiornamento delle statistiche subito prima

di recuperare il riepilogo. Se il grafo cambia durante il calcolo delle statistiche, queste subiranno necessariamente un leggero ritardo, ma non di molto.

Utilizzo dell'API di riepilogo del grafo per recuperare le informazioni di riepilogo del grafo

Per un grafo di proprietà su cui si esegue una query utilizzando Gremlin o openCypher, è possibile recuperare un riepilogo del grafo dall'endpoint di riepilogo del grafo di proprietà. Per questo endpoint esiste un URI lungo e uno breve:

- `https://`*`your-neptune-host`*`:`*`port`*`/propertygraph/statistics/summary`
- `https://`*`your-neptune-host`*`:`*`port`*`/pg/statistics/summary`

Per un grafo RDF su cui si esegue una query utilizzando SPARQL, è possibile recuperare un riepilogo del grafo dall'endpoint di riepilogo RDF:

- `https://`*`your-neptune-host`*`:`*`port`*`/rdf/statistics/summary`

Questi endpoint sono di sola lettura e supportano solo un'operazione HTTP `GET`. Se $GRAPH_SUMMARY_ENDPOINT è impostato sull'indirizzo dell'endpoint su cui si vuole eseguire la query, è possibile recuperare i dati di riepilogo utilizzando `curl` e HTTP `GET` come segue:

```
curl -G "$GRAPH_SUMMARY_ENDPOINT"
```

Se non sono disponibili statistiche quando si tenta di recuperare un riepilogo del grafo, la risposta è simile alla seguente:

```
{
  "detailedMessage": "Statistics are not available. Summary can only be generated after
 statistics are available.",
  "requestId": "48c1f788-f80b-b69c-d728-3f6df579a5f6",
  "code": "StatisticsNotAvailableException"
}
```

Parametro di query dell'URL **`mode`** per l'API di riepilogo del grafo

L'API di riepilogo del grafo accetta un parametro di query dell'URL denominato`mode`, che può assumere uno di due valori, vale a dire `basic` (impostazione predefinita) e `detailed`. Per un

grafo RDF, la risposta di riepilogo del grafo in modalità `detailed` contiene un campo aggiuntivo `subjectStructures`. Per un grafo di proprietà, la risposta di riepilogo dettagliata del grafo contiene due campi aggiuntivi, vale a dire `nodeStructures` e `edgeStructures`.

Per richiedere una risposta di riepilogo del grafo `detailed`, includere il parametro `mode`come segue:

```
curl -G "$GRAPH_SUMMARY_ENDPOINT?mode=detailed"
```

Se il parametro `mode` non è presente, per impostazione predefinita viene utilizzata la modalità `basic`, quindi, sebbene sia possibile specificare `?mode=basic` in modo esplicito, ciò non è necessario.

Risposta di riepilogo di un grafo di proprietà (PG)

Per un grafo di proprietà vuoto, la risposta di riepilogo dettagliata del grafo è la seguente:

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-01-10T07:58:47.972Z",
    "graphSummary" : {
      "numNodes" : 0,
      "numEdges" : 0,
      "numNodeLabels" : 0,
      "numEdgeLabels" : 0,
      "nodeLabels" : [ ],
      "edgeLabels" : [ ],
      "numNodeProperties" : 0,
      "numEdgeProperties" : 0,
      "nodeProperties" : [ ],
      "edgeProperties" : [ ],
      "totalNodePropertyValues" : 0,
      "totalEdgePropertyValues" : 0,
      "nodeStructures" : [ ],
      "edgeStructures" : [ ]
    }
  }
}
```

Una risposta di riepilogo del grafo di proprietà (PG) contiene i seguenti campi:

- **`status`**: codice HTTP restituito della richiesta. Se la richiesta è riuscita, il codice è 200.

  Per visualizzare un elenco di errori comuni, consulta Errori comuni di riepilogo del grafo.

- **`payload`**
  - **`version`**: versione della risposta di riepilogo del grafo.
  - **`lastStatisticsComputationTime`** : timestamp, in formato ISO 8601, dell'ora in cui Neptune ha calcolato le statistiche per l'ultima volta.
  - **`graphSummary`**
    - **`numNodes`**: numero di nodi nel grafo.
    - **`numEdges`**: numero di archi nel grafo.
    - **`numNodeLabels`**: numero di etichette di nodi distinte nel grafo.
    - **`numEdgeLabels`**: numero di etichette di archi distinte nel grafo.
    - **`nodeLabels`**: elenco di etichette di nodi distinte nel grafo.
    - **`edgeLabels`**: elenco di etichette di archi distinte nel grafo.
    - **`numNodeProperties`**: numero di proprietà di nodi distinte nel grafo.
    - **`numEdgeProperties`**: numero di proprietà di archi distinte nel grafo.
    - **`nodeProperties`**: elenco di proprietà di nodi distinte nel grafo, insieme al numero di nodi in cui viene utilizzata ciascuna proprietà.
    - **`edgeProperties`**: elenco di proprietà di archi distinte nel grafo, insieme al numero di archi in cui viene utilizzata ciascuna proprietà.
    - **`totalNodePropertyValues`**: numero totale di utilizzi di tutte le proprietà di nodi.
    - **`totalEdgePropertyValues`**: numero totale di utilizzi di tutte le proprietà di archi.
    - **`nodeStructures`**: questo campo è presente solo quando nella richiesta è specificato *`mode=detailed`*. Contiene un elenco di strutture di nodi, ognuna delle quali contiene i seguenti campi:
      - **`count`**: numero di nodi con questa struttura specifica.
      - **`nodeProperties`**: elenco delle proprietà dei nodi in questa struttura specifica.
      - **`distinctOutgoingEdgeLabels`**: elenco di etichette di archi in uscita distinte in questa struttura specifica.
    - **`edgeStructures`**: questo campo è presente solo quando nella richiesta è specificato *`mode=detailed`*. Contiene un elenco di strutture di archi, ognuna delle quali contiene i seguenti campi:

- **`count`**: numero di archi con questa struttura specifica.
- **`edgeProperties`**: elenco di proprietà di archi in questa struttura specifica.

Risposta di riepilogo di un grafo RDF

Per un grafo RDF vuoto, la risposta di riepilogo dettagliata del grafo è la seguente:

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-01-10T07:58:47.972Z",
    "graphSummary" : {
      "numDistinctSubjects" : 0,
      "numDistinctPredicates" : 0,
      "numQuads" : 0,
      "numClasses" : 0,
      "classes" : [ ],
      "predicates" : [ ],
      "subjectStructures" : [ ]
    }
  }
}
```

Una risposta di riepilogo del grafo RDF contiene i seguenti campi:

- **`status`**: codice HTTP restituito della richiesta. Se la richiesta è riuscita, il codice è 200.

  Per visualizzare un elenco di errori comuni, consulta Errori comuni di riepilogo del grafo.
- **`payload`**
  - **`version`**: versione della risposta di riepilogo del grafo.
  - **`lastStatisticsComputationTime`** : timestamp, in formato ISO 8601, dell'ora in cui Neptune ha calcolato le statistiche per l'ultima volta.
  - **`graphSummary`**
    - **`numDistinctSubjects`**: numero di soggetti distinti nel grafo.
    - **`numDistinctPredicates`**: numero di predicati distinti nel grafo.
    - **`numQuads`**: numero di quadruple nel grafo.
    - **`numClasses`**: numero di classi nel grafo.

- **`classes`**: elenco di classi nel grafo.
- **`predicates`**: elenco di predicati nel grafo, insieme ai conteggi dei predicati.
- **`subjectStructures`**: questo campo è presente solo quando nella richiesta è specificato *`mode=detailed`*. Contiene un elenco di strutture di soggetti, ognuna delle quali contiene i seguenti campi:
  - **`count`**: numero di occorrenze di questa struttura specifica.
  - **`predicates`**: elenco di predicati presenti in questa struttura specifica.

Risposta di riepilogo di un grafo di proprietà (PG) di esempio

Ecco la risposta di riepilogo dettagliata di un grafo di proprietà che contiene il set di dati delle rotte aeree di un grafo di proprietà di esempio:


```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-03-01T14:35:03.804Z",
    "graphSummary" : {
      "numNodes" : 3748,
      "numEdges" : 51300,
      "numNodeLabels" : 4,
      "numEdgeLabels" : 2,
      "nodeLabels" : [
        "continent",
        "country",
        "version",
        "airport"
      ],
      "edgeLabels" : [
        "contains",
        "route"
      ],
      "numNodeProperties" : 14,
      "numEdgeProperties" : 1,
      "nodeProperties" : [
        {
          "desc" : 3748
        },
        {
```

```
          "code" : 3748
        },
        {
          "type" : 3748
        },
        {
          "country" : 3503
        },
        {
          "longest" : 3503
        },
        {
          "city" : 3503
        },
        {
          "lon" : 3503
        },
        {
          "elev" : 3503
        },
        {
          "icao" : 3503
        },
        {
          "region" : 3503
        },
        {
          "runways" : 3503
        },
        {
          "lat" : 3503
        },
        {
          "date" : 1
        },
        {
          "author" : 1
        }
      ],
      "edgeProperties" : [
        {
          "dist" : 50532
        }
      ],
```

```
      "totalNodePropertyValues" : 42773,
      "totalEdgePropertyValues" : 50532,
      "nodeStructures" : [
        {
          "count" : 3471,
          "nodeProperties" : [
            "city",
            "code",
            "country",
            "desc",
            "elev",
            "icao",
            "lat",
            "lon",
            "longest",
            "region",
            "runways",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [
            "route"
          ]
        },
        {
          "count" : 161,
          "nodeProperties" : [
            "code",
            "desc",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [
            "contains"
          ]
        },
        {
          "count" : 83,
          "nodeProperties" : [
            "code",
            "desc",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [ ]
        },
        {
```

```
          "count" : 32,
          "nodeProperties" : [
            "city",
            "code",
            "country",
            "desc",
            "elev",
            "icao",
            "lat",
            "lon",
            "longest",
            "region",
            "runways",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [ ]
        },
        {
          "count" : 1,
          "nodeProperties" : [
            "author",
            "code",
            "date",
            "desc",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [ ]
        }
      ],
      "edgeStructures" : [
        {
          "count" : 50532,
          "edgeProperties" : [
            "dist"
          ]
        }
      ]
    }
  }
}
```

Risposta di riepilogo di un grafo RDF di esempio

Ecco la risposta di riepilogo dettagliata di un grafo RDF che contiene il set di dati delle rotte aeree di un grafo RDF di esempio:


```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-03-01T14:54:13.903Z",
    "graphSummary" : {
      "numDistinctSubjects" : 54403,
      "numDistinctPredicates" : 19,
      "numQuads" : 158571,
      "numClasses" : 4,
      "classes" : [
        "http://kelvinlawrence.net/air-routes/class/Version",
        "http://kelvinlawrence.net/air-routes/class/Airport",
        "http://kelvinlawrence.net/air-routes/class/Continent",
        "http://kelvinlawrence.net/air-routes/class/Country"
      ],
      "predicates" : [
        {
          "http://kelvinlawrence.net/air-routes/objectProperty/route" : 50656
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/dist" : 50656
        },
        {
          "http://kelvinlawrence.net/air-routes/objectProperty/contains" : 7004
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/code" : 3747
        },
        {
          "http://www.w3.org/2000/01/rdf-schema#label" : 3747
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/type" : 3747
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/desc" : 3747
        },
```

```
        {
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" : 3747
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/icao" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/lat" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/region" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/runways" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/longest" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/elev" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/lon" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/country" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/city" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/author" : 1
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/date" : 1
        }
      ],
      "subjectStructures" : [
        {
          "count" : 50656,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/dist"
          ]
        },
```

```
        {
          "count" : 3471,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/city",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/country",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/elev",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/icao",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lat",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lon",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/longest",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/region",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/runways",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://kelvinlawrence.net/air-routes/objectProperty/route",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 238,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://kelvinlawrence.net/air-routes/objectProperty/contains",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 31,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/city",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/country",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/elev",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/icao",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lat",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lon",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/longest",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/region",
```

```
            "http://kelvinlawrence.net/air-routes/datatypeProperty/runways",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 6,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 1,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/author",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/date",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        }
      ]
    }
  }
}
```

Utilizzo dell'autenticazione AWS Identity and Access Management (IAM) con endpoint di riepilogo grafico

È possibile accedere agli endpoint di riepilogo del grafo in modo sicuro con l'autenticazione IAM utilizzando awscurl o qualsiasi altro strumento che funzioni con HTTPS e IAM. Per vedere come configurare le credenziali corrette, consulta Utilizzo di `awscurl` con credenziali temporanee per connettersi in modo sicuro a un cluster database con autenticazione IAM abilitata. Dopo aver eseguito questa operazione, è possibile effettuare richieste come questa:

```
awscurl "$GRAPH_SUMMARY_ENDPOINT" \
    --region (your region) \
    --service neptune-db
```

> **Important**
>
> L'identità o il ruolo IAM che crea le credenziali temporanee deve avere una policy IAM allegata che consenta l'azione GetGraphSummaryIAM.

Per un elenco di errori IAM comuni che si possono riscontrare, consulta Errori di autenticazione IAM.

Codici di errore comuni che possono essere restituiti da una richiesta di riepilogo del grafo

Codice di errore del servizio Neptune	Stato HTT	Messaggio	Scenario di errore	Mitigazione
AccessDeniedException	403	Token di autenticazione mancante.	Una richiesta non firmata o firmata in modo errato è stata inviata al database Neptune con IAM abilitato.	Firmare la richiesta con SigV4 prima dell'invio (vedi IAM e riepiloghi del grafo).
	403	*L'utente: (user ARN) non è autorizzato a eseguire: neptune-db: on GetGraphSummary resource: (resource ARN).*	La policy IAM non consente l'azione GetGraphSummaryquando la richiesta di riepilogo del grafico è stata inviata al database Neptune con IAM abilitato.	Assicurarsi che la policy IAM collegata all'utente o al ruolo che effettua la richiesta consenta l'azione `GetGraphSummary` .
BadRequestException	400	Le statistiche sono disabilitate, quindi	Tentativo di recuperare il	Utilizzare un tipo di istanza in cui è

Codice di errore del servizio Neptune	Stato HTTP	Messaggio	Scenario di errore	Mitigazione
		anche il riepilogo del grafo è disabilitato.	riepilogo sui tipi di istanze espandibili (`t3` o `t4g`) in cui le statistiche sono disabilitate.	abilitata la generazione delle statistiche (tutte le istanze supportate tranne `t3` e `t4g`).
	400	Percorso errato: *`/rdf/statistics/summarypathapi`*	Richiesta inviata a un percorso non valido.	Usare il percorso corretto per l'endpoint di riepilogo del grafo.
`InvalidParameterException`	400	La richiesta contiene parametri sconosciuti: '*`(parametro o parametri sconosciuti)`* '.	Quando nella richiesta viene specificato un parametro non valido.	Utilizzare solo parametri validi (ad esempio, `mode`) nella richiesta.
`InvalidParameterException`	400	Il parametro di query URI 'mode' ha un valore non supportato '*`(valore non valido)`*'.	Quando il parametro URL 'mode' nella richiesta è seguito da un valore non valido.	Utilizzare valori validi (come `basic` o `detailed`) quando si specifica il parametro URL 'mode'.
`MethodNotAllowedException`	405	Metodo non consentito.	Chiamata dell'endpoint di riepilogo con un metodo HTTP diverso da `GET` (come `POST` o `DELETE`).	Usare il metodo HTTP `GET` quando si chiama l'endpoint di riepilogo.

Codice di errore del servizio Neptune	Stato HTTP	Messaggio	Scenario di errore	Mitigazione
StatisticsNotAvailableException	400	Le statistiche non sono ancora state calcolate, il riepilogo del grafo sarà disponibile dopo il completamento del calcolo delle statistiche.	Non ci sono statistiche disponibili quando la richiesta viene inviata all'endpoint di riepilogo.	Attendere il completamento della generazione delle statistiche. È possibile controllare lo stato della generazione delle statistiche utilizzando l'API di stato delle statistiche.
	400	È stato raggiunto il limite delle statistiche, quindi il riepilogo del grafo non è disponibile.	La generazione delle statistiche è stata interrotta perché ha raggiunto i limiti di dimensione delle statistiche.	Il riepilogo del grafo non è disponibile per questo grafo.

Ad esempio, se si effettua una richiesta all'endpoint di riepilogo del grafo in un database Neptune con l'autenticazione IAM abilitata e le autorizzazioni necessarie non sono presenti nella policy IAM del richiedente, si otterrà una risposta simile alla seguente:

```
{
  "detailedMessage": "User: arn:aws:iam::(account ID):(user or user name) is not
 authorized to perform: neptune-db:GetGraphSummary on resource: arn:aws:neptune-
db:(region):(account ID):(cluster resource ID)/*",
  "requestId": "7ac2b98e-b626-d239-1d05-74b4c88fce82",
  "code": "AccessDeniedException"
}
```

Connettività JDBC di Amazon Neptune

Amazon Neptune ha rilasciato un driver JDBC open source che supporta le query openCypher, Gremlin, SQL-Gremlin e SPARQL. La connettività JDBC semplifica la connessione a Neptune con strumenti di business intelligence (BI) come Tableau. L'utilizzo del driver JDBC con Neptune non comporta costi aggiuntivi: si paga comunque solo per le risorse Neptune utilizzate.

Il driver è compatibile con JDBC 4.2 e richiede almeno Java 8. Per ulteriori informazioni su come usare un driver JDBC, consulta la documentazione dell'API JDBC.

Il GitHub progetto, in cui è possibile presentare problemi e aprire richieste di funzionalità, contiene una documentazione dettagliata per il driver:

Driver JDBC per Amazon Neptune

- Uso di SQL con il driver JDBC
- Uso di Gremlin con il driver JDBC
- Uso di openCypher con il driver JDBC
- Uso di SPARQL con il driver JDBC

Guida introduttiva al driver JDBC Neptune

Per utilizzare il driver JDBC Neptune per connettersi a un'istanza Neptune, il driver JDBC deve essere implementato in un'istanza Amazon EC2 nello stesso VPC del cluster database Neptune oppure l'istanza deve essere disponibile tramite un tunnel SSH o un sistema di bilanciamento del carico. Un tunnel SSH può essere configurato internamente nel driver o può essere configurato esternamente.

È possibile scaricare il driver qui. Il driver viene fornito in un pacchetto come singolo file JAR con un nome simile a `neptune-jdbc-1.0.0-all.jar`. Per utilizzarlo, posizionare il file JAR nel `classpath` dell'applicazione. Oppure, se l'applicazione utilizza Maven o Gradle, è possibile utilizzare i comandi Maven o Gradle appropriati per installare il driver dal file JAR.

Il driver necessita di un URL di connessione JDBC per connettersi a Neptune, in un formato simile al seguente:

```
jdbc:neptune:(connection
 type)://(host);property=value;property=value;...;property=value
```

Le sezioni relative a ciascun linguaggio di query del GitHub progetto descrivono le proprietà che è possibile impostare nell'URL di connessione JDBC per quel linguaggio di query.

Se il file JAR è presente nel `classpath` dell'applicazione, non sono necessarie altre configurazioni. È possibile connettere il driver utilizzando l'interfaccia JDBC `DriverManager` e una stringa di connessione Neptune. Ad esempio, se il cluster database Neptune è accessibile tramite l'endpoint `neptune-example.com` sulla porta 8182, è possibile connettersi con openCypher in questo modo:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

void example() {
    String url = "jdbc:neptune:opencypher://bolt://neptune-example:8182";

    Connection connection = DriverManager.getConnection(url);
    Statement statement = connection.createStatement();

    connection.close();
}
```

Le sezioni della documentazione del GitHub progetto per ogni linguaggio di query descrivono come costruire la stringa di connessione quando si utilizza quel linguaggio di query.

Utilizzo di Tableau con il driver JDBC Neptune

Per utilizzare Tableau con il driver JDBC Neptune, iniziare scaricando e installando la versione più recente di Tableau Desktop. Scaricare il file JAR per il driver JDBC Neptune e anche il file del connettore Neptune Tableau (un file `.taco`).

Per connettersi a Tableau per Neptune su un Mac

1. Inserire il file JAR del driver JDBC Neptune nella cartella `/Users/`*`(your user name)`*`/Library/Tableau/Drivers`.
2. Inserire il file `.taco` del connettore Neptune Tableau nella cartella `/Users/`*`(your user name)`*`/Documents/My Tableau Repository/Connectors`.
3. Se è abilitata l'autenticazione IAM, configurare l'ambiente corrispondente. Tenere presente che le variabili di ambiente impostate in `.zprofile/`, `.zshenv/`, `.bash_profile` e così via non funzioneranno. Le variabili di ambiente devono essere impostate in modo da poter essere caricate da un'applicazione GUI.

Un modo per impostare le credenziali consiste nell'inserire la chiave di accesso e la chiave segreta nel file `/Users/(your user name)/.aws/credentials`.

Un modo semplice per impostare la regione del servizio consiste nell'aprire un terminale e immettere il seguente comando, utilizzando la regione dell'applicazione (ad esempio, `us-east-1`):

```
launchctl setenv SERVICE_REGION region name
```

Esistono altri modi per impostare variabili di ambiente che persistano dopo un riavvio, ma qualunque sia la tecnica utilizzata, è necessario impostare variabili accessibili a un'applicazione GUI.

4. Per caricare le variabili di ambiente in una GUI sul Mac, immettere questo comando su un terminale:

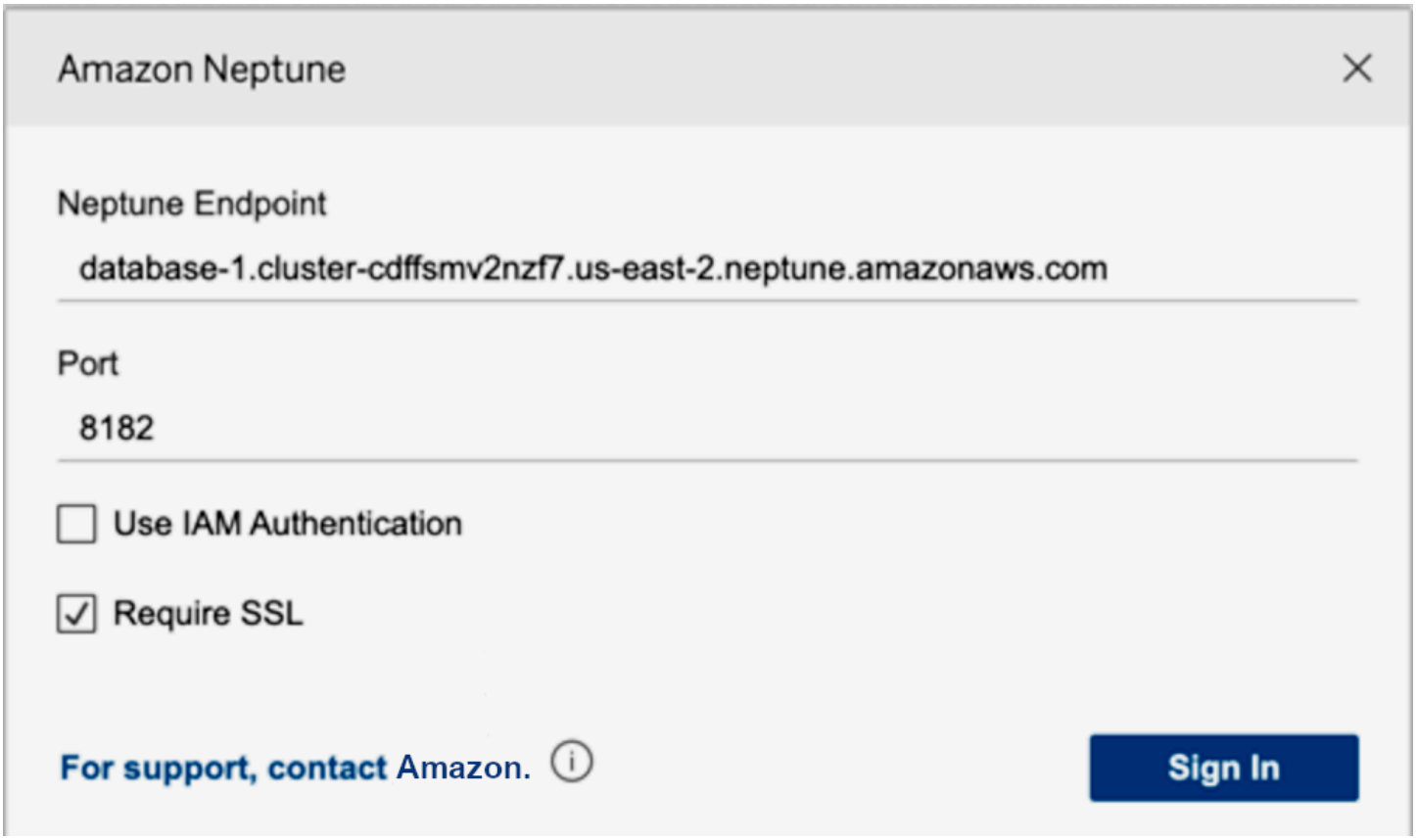
```
/Applications/Tableau/Desktop/2021.1.app/Contents/MacOS/Tableau
```

Per connettersi a Tableau per Neptune su un computer Windows

1. Inserire il file JAR del driver JDBC Neptune nella cartella `C:\Program Files\Tableau\Drivers`.
2. Inserire il file `.taco` del connettore Neptune Tableau nella cartella `C:\Users\(your user name)\Documents\My Tableau Repository\Connectors`.
3. Se è abilitata l'autenticazione IAM, configurare l'ambiente corrispondente.

   Questa operazione può essere semplice come impostare le variabili di ambiente utente `ACCESS_KEY`, `SECRET_KEY` e `SERVICE_REGION`.

Con Tableau aperto, seleziona Altro sul lato sinistro della finestra. Se il file del connettore Tableau è posizionato correttamente, è possibile selezionare Amazon Neptune by AWS nell'elenco visualizzato:



Non dovrebbe essere necessario modificare la porta o aggiungere opzioni di connessione. Immettere l'endpoint Neptune e impostare la configurazione IAM e SSL (è necessario abilitare SSL se si utilizza IAM).

Quando si seleziona Accedi, potrebbero essere necessari più di 30 secondi per la connessione se il grafo è di grandi dimensioni. Tableau raccoglie tabelle di vertici e archi e unisce i vertici agli archi, oltre a creare visualizzazioni.

Risoluzione dei problemi di connessione del driver JDBC

Se il driver non riesce a connettersi al server, utilizzare la funzione `isValid` dell'oggetto JDBC `Connection` per verificare se la connessione è valida. Se la funzione restituisce `false`, per indicare che la connessione non è valida, verificare che l'endpoint a cui ci si sta connettendo sia corretto e che ci si trovi nel VPC del cluster database Neptune o che si disponga di un tunnel SSH valido per il cluster.

Se si ottiene una risposta `No suitable driver found for` *`(connection string)`* dalla chiamata `DriverManager.getConnection`, probabilmente c'è un problema all'inizio della stringa di connessione. Assicurarsi che la stringa di connessione inizi così:

```
jdbc:neptune:opencypher://...
```

Per raccogliere ulteriori informazioni sulla connessione, è possibile aggiungere un oggetto `LogLevel` alla stringa di connessione in questo modo:

```
jdbc:neptune:opencypher://(JDBC URL):(port);logLevel=trace
```

In alternativa, è possibile aggiungere `properties.put("logLevel", "trace")` nelle proprietà di input per registrare le informazioni di traccia.

Aggiornamenti del motore Amazon Neptune

Gli aggiornamenti dei rilasci del motore Amazon Neptune vengono resi disponibili periodicamente. Puoi stabilire quale versione del rilascio del motore hai installato attualmente utilizzando l'API instance-status.

Le versioni del motore sono elencate in Rilasci del motore per Amazon Neptune e le patch sono elencate in Aggiornamenti più recenti.

Per ulteriori informazioni su come vengono rilasciati gli aggiornamenti e su come aggiornare il motore Neptune nel database, consulta Cluster maintenance. Ad esempio, la numerazione delle versioni è descritta in Engine version numbers.

Sicurezza in Amazon Neptune

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che puoi utilizzare in modo sicuro. I revisori di terze parti testano e verificano regolarmente l'efficacia della sicurezza come parte dei programmi di conformitàAWS. Per ulteriori informazioni sui programmi di conformità che si applicano ad Amazon Neptune, consulta Servizi AWS coperti dal programma di conformità.
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. Sei anche responsabile di altri fattori, tra cui la riservatezza dei dati, i requisiti della tua azienda e le leggi e normative vigenti.

Questa documentazione consente di comprendere come applicare il modello di responsabilità condivisa quando si utilizza Neptune. I seguenti argomenti illustrano come configurare Neptune per soddisfare gli obiettivi di sicurezza e conformità. Imparerai anche a usare altri AWS servizi che ti aiutano a monitorare e proteggere le tue risorse di Neptune.

Argomenti

Protezione dei dati in Amazon Neptune

Il modello di responsabilità AWS condivisa Modello si applica alla protezione dei dati in Amazon Neptune. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i. Cloud AWS L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. Inoltre, sei responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS che utilizzi. Per ulteriori informazioni sulla privacy dei dati, vedi le Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al Modello di responsabilità condivisa AWS e GDPR nel Blog sulla sicurezza AWS .

Ai fini della protezione dei dati, consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Usa SSL/TLS per comunicare con le risorse. AWS È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con. AWS CloudTrail
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-2 per l'accesso AWS tramite un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.

Ti consigliamo vivamente di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori con Neptune o Servizi AWS altri utenti utilizzando la console, l'API AWS CLI o gli SDK. AWS I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

Utilizzi chiamate API AWS pubblicate per gestire Neptune attraverso la rete. I client devono supportare Transport Layer Security (TLS) 1.2 o versioni successive utilizzando suite di crittografia avanzate, come descritto in Crittografia in transito. La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Le seguenti sezioni descrivono ulteriormente come vengono protetti i dati di Neptune.

Argomenti

Ogni cluster database Amazon Neptune risiede in un Amazon VPC

Un cluster database Amazon Neptune può essere creato solo in un Amazon Virtual Private Cloud (Amazon VPC) e i relativi endpoint sono accessibili solo all'interno di tale VPC, in genere da un'istanza Amazon Elastic Compute Cloud (Amazon EC2) in esecuzione in tale VPC.

È possibile proteggere i dati di Neptune limitando l'accesso al VPC in cui si trova il cluster database Neptune, come descritto in Connessione al grafo Amazon Neptune.

Crittografia in transito: connessione a Neptune utilizzando il protocollo SSL/HTTPS

A partire dalla versione 1.0.4.0 del motore, Amazon Neptune consente solo connessioni Secure Sockets Layer (SSL) tramite HTTPS a qualsiasi istanza o endpoint del cluster.

Neptune richiede TLS versione 1.2, che utilizza le seguenti suite di crittografia avanzate:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`

Anche laddove le connessioni HTTP sono consentite nelle versioni precedenti del motore, qualsiasi cluster database che utilizza un nuovo gruppo di parametri del cluster database deve utilizzare SSL per impostazione predefinita. Per proteggere i dati, gli endpoint Neptune nella versione del motore

1.0.4.0 e successive supportano solo le richieste HTTPS. Per ulteriori informazioni, consulta Utilizzo dell'endpoint HTTP REST per connettersi a un'istanza database Neptune.

Neptune fornisce automaticamente certificati SSL per le istanze database Neptune. Non è necessario richiedere i certificati. I certificati vengono forniti al momento della creazione di una nuova istanza.

Neptune assegna un singolo certificato SSL wildcard alle istanze del tuo account per ogni regione. AWS Il certificato fornisce voci per gli endpoint del cluster, gli endpoint di sola lettura del cluster e gli endpoint dell'istanza.

Dettagli del certificato

Le seguenti voci sono incluse nel certificato fornito:

- Endpoint del cluster: `*.cluster-`*`a1b2c3d4wxyz`*`.`*`region`*`.neptune.amazonaws.com`
- Endpoint di sola lettura: `*.cluster-ro-`*`a1b2c3d4wxyz`*`.`*`region`*`.neptune.amazonaws.com`
- Endpoint dell'istanza: `*.`*`a1b2c3d4wxyz`*`.`*`region`*`.neptune.amazonaws.com`

Sono supportate solo le voci elencate in questa pagina.

Connessioni proxy

I certificati supportano solo i nomi host elencati nella sezione precedente.

Se usi un sistema di bilanciamento del carico o un server proxy (come HAProxy), è necessario utilizzare la terminazione SSL e avere un certificato SSL sul server proxy.

Il passthrough SSL non funziona perché i certificati SSL forniti non corrispondono al nome host del server proxy.

Certificati CA radice

I certificati per le istanze Neptune vengono normalmente convalidate utilizzando l'archivio di trust locale del sistema operativo o dell'SDK (come Java SDK).

Se devi fornire un certificato radice manualmente, puoi scaricare il certificato CA radice di Amazon in formato PEM dal repository delle policy di Amazon Trust Services.

Ulteriori informazioni

Per ulteriori informazioni sulla connessione agli endpoint Neptune con SSL, consulta the section called “Installazione della console Gremlin” e the section called “HTTP REST”.

Crittografia delle risorse Neptune inattive

Le istanze crittografate di Neptune offrono un livello aggiuntivo di protezione dei dati proteggendoli dagli accessi non autorizzati all'archiviazione sottostante. Puoi usare la crittografia Neptune per aumentare la protezione dei dati delle applicazioni implementate nel cloud. Puoi utilizzarlo anche per soddisfare i requisiti di conformità per la crittografia. data-at-rest

Per gestire le chiavi utilizzate per crittografare e decrittografare le risorse di Neptune, si utilizza ().AWS Key Management ServiceAWS KMS AWS KMS combina hardware e software sicuri e ad alta disponibilità per fornire un sistema di gestione delle chiavi scalabile per il cloud. Utilizzando AWS KMS, è possibile creare chiavi di crittografia e definire le politiche che controllano il modo in cui tali chiavi possono essere utilizzate. AWS KMS supporta AWS CloudTrail, in modo da poter controllare l'utilizzo delle chiavi per verificare che vengano utilizzate in modo appropriato. Puoi utilizzare AWS KMS le tue chiavi in combinazione con Neptune e servizi AWS supportati come Amazon Simple Storage Service (Amazon S3), Amazon Elastic Block Store (Amazon EBS) e Amazon Redshift. Per un elenco dei servizi che supportano AWS KMS, consulta How AWS Services AWS KMS Use nella Developer Guide.AWS Key Management Service

Tutti i log, i backup e gli snapshot sono crittografati per un'istanza Neptune crittografata.

Abilitazione della crittografia per un'istanza database Neptune

Per abilitare la crittografia per una nuova istanza database Neptune, scegliere Sì nella sezione Abilita crittografia della console Neptune. Per informazioni sulla creazione di un'istanza database Neptune, consulta Creazione di un nuovo cluster database Neptune.

Quando crei un'istanza crittografata di Neptune DB, puoi anche fornire AWS KMS l'identificatore della chiave per la tua chiave di crittografia. Se non specifichi un identificatore di AWS KMS chiave, Neptune utilizza la chiave di crittografia Amazon RDS predefinita `aws/rds` () per la tua nuova istanza DB Neptune. AWS KMS crea la chiave di crittografia predefinita per Neptune per il tuo account. AWS Il tuo AWS account ha una chiave di crittografia predefinita diversa per ogni AWS regione.

Dopo aver creato un'istanza database Neptune crittografata, non è possibile modificare la chiave di crittografia per quell'istanza. Quindi, assicurati di determinare i requisiti della chiave di crittografia prima di creare l'istanza database Neptune crittografata.

È possibile utilizzare il nome della risorsa Amazon (ARN) di una chiave di un altro account per crittografare un'istanza database Neptune. Se crei un'istanza di Neptune DB con AWS lo stesso account che possiede AWS KMS la chiave di crittografia utilizzata per crittografare quella nuova

istanza di Neptune DB AWS KMS , l'ID della chiave che passi può AWS KMS essere l'alias della chiave anziché l'ARN della chiave.

> **Important**
>
> Se Neptune perde l'accesso alla chiave di crittografia per un'istanza database Neptune, ad esempio quando viene revocato l'accesso di Neptune a una chiave, l'istanza database crittografata viene impostata su uno stato terminale e può essere ripristinata solo da un backup. È consigliabile sempre abilitare i backup per le istanze database Neptune crittografate per evitare la perdita di dati crittografati nei database.

Autorizzazioni chiave necessarie per abilitare la crittografia

L'utente o il ruolo IAM che crea un'istanza database Neptune crittografata deve disporre almeno delle seguenti autorizzazioni per la chiave KMS:

- `"kms:Encrypt"`
- `"kms:Decrypt"`
- `"kms:GenerateDataKey"`
- `"kms:ReEncryptTo"`
- `"kms:GenerateDataKeyWithoutPlaintext"`
- `"kms:CreateGrant"`
- `"kms:ReEncryptFrom"`
- `"kms:DescribeKey"`

Ecco un esempio di una policy della chiave che include le autorizzazioni necessarie:

```
{
  "Version": "2012-10-17",
  "Id": "key-consolepolicy-3",
  "Statement": [
    {
      "Sid": "Enable Permissions for root principal",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
```

```
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow use of the key for Neptune",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/NeptuneFullAccess"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:ReEncryptTo",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:CreateGrant",
        "kms:ReEncryptFrom",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "rds.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "Deny use of the key for non Neptune",
      "Effect": "Deny",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/NeptuneFullAccess"
      },
      "Action": [
        "kms:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "kms:ViaService": "rds.us-east-1.amazonaws.com"
        }
      }
    }
  ]
```

```
}
```

- La prima istruzione di questa policy è facoltativa. Fornisce accesso al principale root dell'utente.
- La seconda istruzione fornisce l'accesso a tutte le AWS KMS API richieste per questo ruolo, limitatamente all'RDS Service Principal.
- La terza dichiarazione rafforza ulteriormente la sicurezza imponendo che questa chiave non sia utilizzabile da questo ruolo per nessun altro servizio. AWS

È anche possibile ridurre ulteriormente l'ambito delle autorizzazioni `createGrant` aggiungendo:

```
"Condition": {
  "Bool": {
    "kms:GrantIsForAWSResource": true
  }
}
```

Limitazioni della crittografia Neptune

Alla crittografia dei cluster Neptune si applicano le seguenti limitazioni:

- Non è possibile convertire un cluster database non crittografato in uno crittografato.

  Tuttavia, puoi ripristinare un cluster database non crittografato in un cluster database crittografato. Per eseguire questa operazione, specifica una chiave di crittografia KMS quando ripristini dallo snapshot di cluster database non crittografato.
- Non è possibile convertire un'istanza database non crittografata in una crittografata. È possibile abilitare la crittografia di un'istanza database solo al momento della creazione.
- Inoltre, le istanze database crittografate non possono essere modificate per disabilitare la crittografia.
- Non è possibile creare una replica di lettura crittografata di un'istanza database non crittografata o una replica di lettura non crittografata di un'istanza database crittografata.
- Le repliche di lettura crittografate devono essere crittografate con la stessa chiave dell'istanza database dell'origine.

Panoramica di AWS Identity and Access Management (IAM) in Amazon Neptune

AWS Identity and Access Management (IAM) è un programma Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle risorse. AWS Gli amministratori IAM controllano chi può essere autenticato (accedere) e autorizzato (disporre delle autorizzazioni) per utilizzare le risorse Neptune. IAM è un software Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Puoi usare AWS Identity and Access Management (IAM) per autenticarti sull'istanza o sul cluster DB di Neptune. Quando l'autenticazione del database IAM è abilitata, ogni richiesta deve essere firmata utilizzando AWS Signature Version 4.

AWS Signature Version 4 aggiunge informazioni di autenticazione alle AWS richieste. Per motivi di sicurezza, tutte le richieste ai cluster database Neptune con autenticazione IAM abilitata devono essere firmate con una chiave di accesso. La chiave è composta da un ID chiave di accesso e una chiave di accesso segreta. L'autenticazione viene gestita esternamente utilizzando le policy IAM.

Neptune si autentica durante la connessione e WebSockets per le connessioni verifica periodicamente le autorizzazioni per garantire che l'utente abbia ancora accesso.

> **Note**
>
> - La revoca, l'eliminazione o la rotazione delle credenziali associate all'utente IAM non è consigliata perché non interrompe le connessioni già aperte.
> - Esistono limiti al numero di WebSocket connessioni simultanee per istanza di database e al tempo in cui una connessione può rimanere aperta. Per ulteriori informazioni, consulta WebSockets Limiti.

L'uso di IAM dipende dal ruolo

Il modo in cui usi AWS Identity and Access Management (IAM) varia a seconda del lavoro svolto in Neptune.

Utente del servizio: se utilizzi il servizio Neptune per eseguire il tuo processo, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie per utilizzare il piano dati di Neptune. Quando

hai bisogno di un accesso maggiore per svolgere il tuo lavoro, capire come viene gestito l'accesso ai dati ti consente di richiedere le autorizzazioni corrette all'amministratore.

Amministratore del servizio: se sei il responsabile delle risorse Neptune presso la tua azienda, probabilmente disponi dell'accesso alle operazioni di gestione di Neptune, che corrispondono all'API di gestione Neptune. Il tuo compito potrebbe anche essere quello di determinare quali azioni di accesso ai dati e risorse Neptune sono necessarie agli utenti del servizio per svolgere il proprio lavoro. Un amministratore IAM può quindi applicare le policy IAM per modificare le autorizzazioni degli utenti del servizio.

Amministratore IAM: se sei un amministratore IAM, dovrai scrivere policy IAM per gestire sia la gestione che l'accesso ai dati di Neptune. Per visualizzare esempi di policy basate su identità di Neptune che puoi utilizzare, consulta Utilizzo di diversi tipi di policy IAM per controllare l'accesso a Neptune.

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi essere autenticato (aver effettuato l' Utente root dell'account AWS accesso AWS) come utente IAM o assumendo un ruolo IAM.

Puoi accedere AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS IAM Identity Center Gli utenti (IAM Identity Center), l'autenticazione Single Sign-On della tua azienda e le tue credenziali di Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Quando accedi AWS utilizzando la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere al AWS Management Console o al portale di AWS accesso. Per ulteriori informazioni sull'accesso a AWS, vedi Come accedere al tuo Account AWS nella Guida per l'Accedi ad AWS utente.

Se accedi a AWS livello di codice, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le tue richieste utilizzando le tue credenziali. Se non utilizzi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sull'utilizzo del metodo consigliato per firmare autonomamente le richieste, consulta Signing AWS API request nella IAM User Guide.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, ti AWS consiglia di utilizzare l'autenticazione a più fattori

(MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'utente di AWS IAM Identity Center e Utilizzo dell'autenticazione a più fattori (MFA) in AWS nella Guida per l'utente di IAM.

Account AWS utente root

Quando si crea un account Account AWS, si inizia con un'identità di accesso che ha accesso completo a tutte Servizi AWS le risorse dell'account. Questa identità è denominata utente Account AWS root ed è accessibile effettuando l'accesso con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzarle per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione Attività che richiedono le credenziali dell'utente root nella Guida per l'utente di IAM.

Utenti e gruppi IAM

Un utente IAM è un'identità interna Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, per casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente di IAM.

Un gruppo IAM è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile avere un gruppo denominato Amministratori IAM e concedere a tale gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Quando creare un utente IAM (invece di un ruolo) nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. Puoi assumere temporaneamente un ruolo IAM in AWS Management Console cambiando ruolo. Puoi assumere un ruolo chiamando un'operazione AWS CLI o AWS API o utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente di IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato: per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Creazione di un ruolo per un provider di identità di terza parte nella Guida per l'utente di IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .
- Autorizzazioni utente IAM temporanee: un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, con alcuni Servizi AWS, è possibile allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.
- Accesso a più servizi: alcuni Servizi AWS utilizzano le funzionalità di altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è comune che tale servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
  - Sessioni di accesso inoltrato (FAS): quando utilizzi un utente o un ruolo IAM per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra azione in un servizio diverso. FAS utilizza le autorizzazioni del principale che chiama an Servizio AWS, combinate con la richiesta Servizio AWS per

effettuare richieste ai servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Forward access sessions.

- Ruolo di servizio: un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire azioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.
- Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un ruolo IAM per gestire le credenziali temporanee per le applicazioni in esecuzione su un'istanza EC2 e che AWS CLI effettuano richieste API. AWS Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un AWS ruolo a un'istanza EC2 e renderlo disponibile per tutte le sue applicazioni, crei un profilo di istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2 nella Guida per l'utente di IAM.

Per informazioni sull'utilizzo dei ruoli IAM, consulta Quando creare un ruolo IAM (invece di un utente) nella Guida per l'utente di IAM.

Abilitazione dell'autenticazione del database IAM in Neptune

Per impostazione predefinita, l'autenticazione database IAM è disabilitata al momento della creazione di un cluster database Amazon Neptune. Puoi attivare l'autenticazione database IAM (o disattivarla nuovamente) utilizzando la AWS Management Console.

Per creare un nuovo cluster database Neptune con l'autenticazione IAM utilizzando la console, segui le istruzioni per la creazione di un cluster database Neptune nell'argomento Avvio di un cluster database Neptune mediante la console AWS Management Console.

Nella seconda pagina del processo di creazione, per Enable IAM DB Authentication (Attiva autenticazione DB IAM) scegli Yes (Sì).

Per abilitare o disabilitare l'autenticazione IAM per un'istanza o un cluster database esistente

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel pannello di navigazione scegliere Clusters (Cluster).
3. Scegli il cluster database Neptune che desideri modificare e scegli Azioni cluster. Scegli Modify cluster (Modifica cluster).
4. Nella sezioneDatabase options (Opzioni database), per IAM DB Authentication (Autenticazione DB IAM) scegli Enable IAM DB authorization (Attiva autorizzazione DB IAM) o No (per disattivare). Quindi, scegliere Continue (Continua).
5. Per applicare immediatamente le modifiche, scegliere Apply immediately (Applica immediatamente).
6. Scegliere Modify cluster (Modifica cluster).

Connessione e firma con AWS Signature versione 4

Le risorse Amazon Neptune con autenticazione IAM DB abilitata richiedono che tutte le richieste HTTP siano firmate AWS utilizzando Signature Version 4. Per informazioni generali sulla firma delle richieste con AWS Signature Version 4, consulta Signing AWS API request.

AWS Signature Version 4 è il processo per aggiungere informazioni di autenticazione alle AWS richieste. Per motivi di sicurezza, la maggior parte delle richieste AWS deve essere firmata con una chiave di accesso, che consiste in un ID della chiave di accesso e una chiave di accesso segreta.

> **Note**
>
> Se utilizzi credenziali temporanee, queste scadono dopo un intervallo specificato, comprensivo del token di sessione.
> Devi aggiornare il token di sessione quando richiedi le nuove credenziali. Per ulteriori informazioni, vedere Utilizzo di credenziali di sicurezza temporanee per richiedere l'accesso alle AWS risorse.

> **Important**
>
> L'accesso a Neptune con l'autenticazione basata su IAM richiede la creazione di richieste HTTP e la firma personale delle richieste stesse.

Come funziona Signature Version 4

1. Si crea una richiesta canonica.
2. Utilizzi la richiesta canonica e alcune altre informazioni per creare un. string-to-sign
3. Utilizzi la tua chiave di accesso AWS segreta per derivare una chiave di firma, quindi usi quella chiave di firma e poi string-to-sign per creare una firma.
4. Si aggiunge la firma risultante alla richiesta HTTP in un'intestazione o come parametro della stringa di query.

Quando Neptune riceve la richiesta, esegue le stesse operazioni che sono state effettuate per calcolare la firma. Neptune quindi confronta la firma calcolata con quella inviata insieme alla richiesta. Se le firme corrispondono, la richiesta viene elaborata. In caso contrario, viene rifiutata.

Per informazioni generali sulla firma delle richieste con AWS Signature Version 4, vedere Processo di firma della versione 4 di Signature in. Riferimenti generali di AWS

Le sezioni successive illustrano esempi di come inviare le richieste firmate agli endpoint Gremlin e SPARQL di un'istanza database Neptune con l'autenticazione IAM abilitata.

Argomenti

- Prerequisiti su Amazon Linux EC2
- Utilizzo di uno strumento a riga di comando per inviare query al cluster database Neptune
- Connessione a Neptune tramite la console Gremlin con la firma di Signature Version 4
- Connessione a Neptune tramite Java e Gremlin con firma di Signature Version 4
- Connessione a Neptune tramite Java e SPARQL con firma di Signature Version 4 (RDF4J e Jena)
- Connessione a Neptune tramite SPARQL e Node.js con firma di Signature Version 4
- Esempio: Connessione a Neptune tramite Python con la firma di Signature Version 4

Prerequisiti su Amazon Linux EC2

Di seguito sono riportate le istruzioni per l'installazione di Apache Maven e Java 8 su un'istanza Amazon EC2. Questi sono necessari per gli esempi di autenticazione di Signature Version 4 di Amazon Neptune.

Installazione di Apache Maven e Java 8 sull'istanza EC2.

1. Collegare l'istanza Amazon EC2 a un client SSH.
2. Installare Apache Maven sull'istanza EC2. Per aggiungere un repository con un pacchetto Maven, immetti prima quanto segue:

   ```
   sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Per impostare il numero di versione per i pacchetti, immetti quanto segue.

   ```
   sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Potrai quindi utilizzare yum per installare Maven.

   ```
   sudo yum install -y apache-maven
   ```
3. Le librerie della console Gremlin richiedono Java 8. Per installare Java 8 sull'istanza EC2, digita quanto segue.

   ```
   sudo yum install java-1.8.0-devel
   ```
4. Per impostare Java 8 come runtime predefinito sull'istanza EC2, digitare quanto segue.

   ```
   sudo /usr/sbin/alternatives --config java
   ```

   Quando richiesto, immetti il numero per Java 8.
5. Per impostare Java 8 come compilatore predefinito sull'istanza EC2, immetti quanto segue:

   ```
   sudo /usr/sbin/alternatives --config javac
   ```

   Quando richiesto, immetti il numero per Java 8.

Utilizzo di uno strumento a riga di comando per inviare query al cluster database Neptune

Avere uno strumento a riga di comando per inviare query al cluster database Neptune è molto utile, come illustrato in molti esempi di questa documentazione. Lo strumento curl è un'opzione eccellente per comunicare con gli endpoint Neptune se l'autenticazione IAM non è abilitata.

Tuttavia, per proteggere i dati è meglio abilitare l'autenticazione IAM.

Quando è abilitata l'autenticazione IAM, ogni richiesta deve essere firmata utilizzando Signature Version 4 (Sig4). Lo strumento a riga di comando awscurl di terze parti utilizza la stessa sintassi di `curl` e può firmare le query utilizzando la firma Sig4. La sezione Uso di `awscurl` seguente spiega come utilizzare `awscurl` in modo sicuro con le credenziali temporanee.

Configurazione di uno strumento a riga di comando per l'utilizzo di HTTPS

Neptune richiede che tutte le connessioni utilizzino HTTPS. Qualsiasi strumento a riga di comando come `curl` o `awscurl` necessita dell'accesso ai certificati appropriati per poter utilizzare HTTPS. Se `curl` o `awscurl` sono in grado di individuare i certificati adeguati, gestiscono le connessioni HTTPS in modo analogo alle connessioni HTTP, senza parametri aggiuntivi. Gli esempi di questa documentazione si basano su questo scenario.

Per scoprire come ottenere tali certificati e come formattarli correttamente in un archivio di certificati CA (Certificate Authority) che `curl` può utilizzare, consulta Verifica certificati SSL nella documentazione di `curl`.

È possibile specificare il percorso di questo archivio di certificati CA utilizzando la variabile di ambiente `CURL_CA_BUNDLE`. In Windows, `curl` cerca automaticamente un file denominato `curl-ca-bundle.crt`. Cerca prima nella stessa cartella di `curl.exe`, quindi in altri punti del percorso. Per ulteriori informazioni, consulta Verifica certificati SSL.

Utilizzo di **`awscurl`** con credenziali temporanee per connettersi in modo sicuro a un cluster database con autenticazione IAM abilitata

Lo strumento awscurl utilizza la stessa sintassi di `curl`, ma necessita anche di informazioni aggiuntive:

- **`--access_key`**: chiave di accesso valida. Se non viene fornita utilizzando questo parametro, deve essere fornita nella variabile di ambiente `AWS_ACCESS_KEY_ID` o in un file di configurazione.

- **`--secret_key`**: chiave segreta valida corrispondente alla chiave di accesso. Se non viene fornita utilizzando questo parametro, deve essere fornita nella variabile di ambiente `AWS_SECRET_ACCESS_KEY` o in un file di configurazione.
- **`--security_token`**: token di sessione valido. Se non viene fornita utilizzando questo parametro, deve essere fornita nella variabile di ambiente `AWS_SECURITY_TOKEN` o in un file di configurazione.

In passato, era prassi comune utilizzare credenziali persistenti con `awscurl`, come le credenziali utente IAM o anche le credenziali root, ma ciò non è consigliato. Generare invece credenziali temporanee utilizzando una delle API AWS Security Token Service (STS) o uno dei relativi wrapper AWS CLI.

È preferibile inserire i valori `AccessKeyId`, `SecretAccessKey` e `SessionToken` restituiti dalla chiamata STS nelle variabili di ambiente appropriate nella sessione della shell anziché in un file di configurazione. In questo modo, quando la shell termina, le credenziali vengono automaticamente eliminate, cosa che non avviene con un file di configurazione. Allo stesso modo, non richiedere per le credenziali temporanee una durata più lunga di quella di cui si potrebbe aver bisogno.

L'esempio seguente mostra i passaggi da eseguire in una shell Linux per ottenere credenziali temporanee valide per mezz'ora usando sts assume-role, per poi inserirle in variabili di ambiente dove `awscurl` può trovarle:

```
aws sts assume-role \
    --duration-seconds 1800 \
    --role-arn "arn:aws:iam::(account-id):role/(rolename)" \
    --role-session-name AWSCLI-Session > $output
AccessKeyId=$(cat $output | jq '.Credentials''.AccessKeyId')
SecretAccessKey=$(cat $output | jq '.Credentials''.SecretAccessKey')
SessionToken=$(cat $output | jq '.Credentials''.SessionToken')

export AWS_ACCESS_KEY_ID=$AccessKeyId
export AWS_SECRET_ACCESS_KEY=$SecretAccessKey
export AWS_SESSION_TOKEN=$SessionToken
```

È possibile quindi usare `awscurl` per effettuare una richiesta firmata al cluster database in questo modo:

```
awscurl (your cluster endpoint):8182/status \
    --region us-east-1 \
```

```
    --service neptune-db
```

Connessione a Neptune tramite la console Gremlin con la firma di Signature Version 4

Il modo in cui ti connetti ad Amazon Neptune utilizzando la console Gremlin con autenticazione Signature versione 4 dipende dal fatto che tu stia TinkerPop utilizzando una versione o una versione successiva o una `3.4.11` versione precedente. In entrambi i casi, sono necessari i seguenti prerequisiti:

- È necessario disporre delle credenziali IAM necessarie per firmare le richieste. Vedi Utilizzo della catena di fornitori di credenziali predefinita nella Guida per gli sviluppatori. AWS SDK for Java
- È necessario aver installato una versione della console Gremlin compatibile con la versione del motore Neptune utilizzata dal cluster database.

Se si utilizzano credenziali temporanee, queste scadono dopo un intervallo specificato, così come il token di sessione, quindi è necessario aggiornare il token di sessione quando si richiedono nuove credenziali. Vedi Utilizzo di credenziali di sicurezza temporanee per richiedere l'accesso alle AWS risorse nella Guida per l'utente IAM.

Per informazioni sulla connessione tramite SSL/TLS, consulta Configurazione SSL/TLS.

Utilizzo della versione TinkerPop 3.4.11 o superiore per connettersi a Neptune con la firma Sig4

Con TinkerPop 3.4.11 o versioni successive, utilizzeraihandshakeInterceptor(), che fornisce un modo per collegare un firmatario Sigv4 alla connessione stabilita dal comando. `:remote` Come per l'approccio utilizzato per Java, è necessario configurare l'oggetto `Cluster` manualmente e quindi passarlo al comando `:remote`.

Tenere presente che questo è molto diverso dalla situazione tipica in cui il comando `:remote` richiede un file di configurazione per formare la connessione. L'approccio del file di configurazione non funzionerà perché `handshakeInterceptor()` deve essere impostato a livello di codice e non può caricare la configurazione da un file.

Connetti la console Gremlin (TinkerPop 3.4.11 e versioni successive) con la firma Sig4

1. Avviare la console Gremlin:

```
$ bin/gremlin.sh
```

2. Al prompt `gremlin>`, installare la libreria `amazon-neptune-sigv4-signer` (questa operazione deve essere eseguita solo una volta per la console):

```
:install com.amazonaws amazon-neptune-sigv4-signer 2.4.0
```

Se riscontri problemi con questo passaggio, può essere utile consultare la documentazione sulla configurazione di Grape. TinkerPop

> **Note**
>
> Se si utilizza un proxy HTTP, è possibile che si verifichino errori in questo passaggio in cui il comando `:install` non viene completato. Per risolvere questo problema, esegui i comandi seguenti per fornire alla console informazioni sul proxy:
>
> ```
> System.setProperty("https.proxyHost", "(the proxy IP address)")
> System.setProperty("https.proxyPort", "(the proxy port)")
> ```

3. Importare la classe richiesta per gestire l'accesso a `handshakeInterceptor()`:

```
:import com.amazonaws.auth.DefaultAWSCredentialsProviderChain
:import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer
```

4. Se si utilizzano credenziali temporanee, è necessario fornire anche il token di sessione come segue:

```
System.setProperty("aws.sessionToken","(your session token)")
```

5. Se non si sono ancora stabilite le credenziali dell'account, è possibile assegnarle come segue:

```
System.setProperty("aws.accessKeyId","(your access key)")
System.setProperty("aws.secretKey","(your secret key)")
```

6. Costruire manualmente l'oggetto `Cluster` da connettere a Neptune:

```
cluster = Cluster.build("(host name)")  \
                .enableSsl(true) \
```

```
                .handshakeInterceptor { r ->  \
                  def sigV4Signer = new NeptuneNettyHttpSigV4Signer("(Amazon
region)", \
                                        new DefaultAWSCredentialsProviderChain()); \
                  sigV4Signer.signRequest(r); \
                  return r; } \
                .create()
```

Per informazioni su come trovare il nome host di un'istanza database Neptune, consulta Connessione agli endpoint Amazon Neptune.

7. Stabilire la connessione `:remote` utilizzando il nome della variabile dell'oggetto `Cluster` nel passaggio precedente:

```
:remote connect tinkerpop.server cluster
```

8. Immettere il seguente comando per passare alla modalità remota. In questo modo tutte le query Gremlin vengono inviate alla connessione remota:

```
:remote console
```

Utilizzo di una versione TinkerPop precedente alla 3.4.11 per connettersi a Neptune con la firma Sig4

Con la TinkerPop versione 3.4.10 o precedente, usa la `amazon-neptune-gremlin-java-sigv4` libreria fornita da Neptune per connettere la console a Neptune con la firma Sigv4, come descritto di seguito:

Connect la console Gremlin (TinkerPop versioni precedenti alla 3.4.11) con la firma Sig4

1. Avviare la console Gremlin:

```
$ bin/gremlin.sh
```

2. Al prompt `gremlin>`, installare la libreria `amazon-neptune-sigv4-signer` (questa operazione deve essere eseguita solo una volta per la console):

```
:install com.amazonaws amazon-neptune-sigv4-signer 2.4.0
```

> **Note**
>
> Se si utilizza un proxy HTTP, è possibile che si verifichino errori in questo passaggio in cui il comando `:install` non viene completato. Per risolvere questo problema, esegui i comandi seguenti per fornire alla console informazioni sul proxy:
>
> ```
> System.setProperty("https.proxyHost", "(the proxy IP address)")
> System.setProperty("https.proxyPort", "(the proxy port)")
> ```
>
> Può essere utile anche consultare la documentazione sulla configurazione di GrapeTinkerPop .

3. Nella sottodirectory `conf` della directory estratta, creare un file denominato `neptune-remote.yaml`.

   Se hai usato il AWS CloudFormation modello per creare il tuo cluster Neptune DB, esisterà già `neptune-remote.yaml` un file. In tal caso, è sufficiente modificare il file esistente per includere l'impostazione del channelizer mostrata di seguito.

   In alternativa, copiare il testo seguente nel file, sostituendo *(nome host)* con il nome host o l'indirizzo IP dell'istanza database Neptune. Tenere presente che le parentesi quadre ([ ]) che racchiudono il nome host sono obbligatorie.

   ```
   hosts: [(host name)]
   port: 8182
   connectionPool: {
     channelizer: org.apache.tinkerpop.gremlin.driver.SigV4WebSocketChannelizer,
     enableSsl: true
   }
   serializer: { className:
    org.apache.tinkerpop.gremlin.driver.ser.GraphBinaryMessageSerializerV1,
                config: { serializeResultToString: true }}
   ```

4. > **Important**
   >
   > È necessario fornire le credenziali IAM per firmare le richieste. Inserisci i seguenti comandi per impostare le credenziali come variabili di ambiente, sostituendo gli elementi rilevanti con le tue credenziali.

```
export AWS_ACCESS_KEY_ID=access_key_id
export AWS_SECRET_ACCESS_KEY=secret_access_key
export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or
 ca-central-1 or
                      sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or
 eu-west-3 or eu-central-1 or me-south-1 or
                      me-central-1 or il-central-1 or af-south-1 or
 ap-east-1 or ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-
southeast-2 or ap-south-1 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

Signer Version 4 di Neptune utilizza la catena di provider di credenziali predefinita. Per conoscere altri metodi per fornire le credenziali, consulta Utilizzo della catena di provider delle credenziali predefinita nella Guida per gli sviluppatori di AWS SDK for Java . La variabile `SERVICE_REGION` è obbligatoria anche quando si utilizza un file delle credenziali.

5. Stabilire la connessione `:remote` utilizzando il file `.yaml`:

```
:remote connect tinkerpop.server conf/neptune-remote.yaml
```

6. Immettere il seguente comando per passare alla modalità remota, che invia tutte le query Gremlin alla connessione remota:

```
:remote console
```

Connessione a Neptune tramite Java e Gremlin con firma di Signature Version 4

Utilizzo della versione TinkerPop 3.4.11 o superiore per connettersi a Neptune con la firma Sig4

Ecco un esempio di come connettersi a Neptune utilizzando l'API Java Gremlin con firma Sig4 quando si utilizza 3.4.11 o versioni successive (si presuppone una conoscenza generale sull' TinkerPop uso di Maven). Innanzitutto, definire le dipendenze come parte del file `pom.xml`:

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>amazon-neptune-sigv4-signer</artifactId>
  <version>2.4.0</version>
</dependency>
```

Quindi, utilizzare un codice come il seguente:

```
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer;
import com.amazonaws.neptune.auth.NeptuneSigV4SignerException;

 ...

System.setProperty("aws.accessKeyId","your-access-key");
System.setProperty("aws.secretKey","your-secret-key");

 ...

Cluster = Cluster.build((your cluster name))
                 .enableSsl(true)
                 .handshakeInterceptor( r ->
                  {
                    try {
                      NeptuneNettyHttpSigV4Signer sigV4Signer =
                        new NeptuneNettyHttpSigV4Signer("(your region)", new
 DefaultAWSCredentialsProviderChain());
                      sigV4Signer.signRequest(r);
                    } catch (NeptuneSigV4SignerException e) {
                      throw new RuntimeException("Exception occurred while signing the
 request", e);
                    }
                    return r;
                  }
                 ).create();
try {
  Client client = cluster.connect();
  client.submit("g.V().has('code','IAD')").all().get();
} catch (Exception e) {
  throw new RuntimeException("Exception occurred while connecting to cluster", e);
}
```

> **Note**
>
> Se si sta eseguendo l'aggiornamento da `3.4.11`, rimuovere i riferimenti alla libreria `amazon-neptune-gremlin-java-sigv4`. Non è più necessaria quando si utilizza `handshakeInterceptor()` come mostrato nell'esempio precedente. Non tentare di utilizzare `handshakeInterceptor()` insieme al channelizer (`SigV4WebSocketChannelizer.class`), perché produrrà errori.

Utilizzo di una versione TinkerPop precedente alla 3.4.11 per connettersi a Neptune con la firma Sig4

TinkerPop le versioni precedenti `3.4.11` non supportavano la `handshakeInterceptor()` configurazione mostrata nella sezione precedente e quindi dovevano fare affidamento sul `amazon-neptune-gremlin-java-sigv4` pacchetto. Questa è una libreria Neptune che contiene `SigV4WebSocketChannelizer` la classe, che sostituisce il Channelizer TinkerPop standard con uno che può iniettare automaticamente una firma SigV4. Ove possibile, aggiorna alla versione 3.4.11 o superiore, perché la libreria è obsoleta TinkerPop . `amazon-neptune-gremlin-java-sigv4`

Ecco un esempio di come connettersi a Neptune utilizzando l'API Java Gremlin con firma Sig4 quando si TinkerPop utilizzano versioni precedenti alla 3.4.11 (si presuppone una conoscenza generale su come usare Maven).

Innanzitutto, definire le dipendenze come parte del file `pom.xml`:

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>amazon-neptune-gremlin-java-sigv4</artifactId>
  <version>2.4.0</version>
</dependency>
```

La dipendenza di cui sopra include la versione `3.4.10` del driver Gremlin. Sebbene sia possibile utilizzare versioni più recenti del driver Gremlin (fino alla `3.4.13`), un aggiornamento del driver dopo la versione 3.4.10 dovrebbe includere una modifica per utilizzare il modello `handshakeInterceptor()` sopra descritto.

L'oggetto Cluster `gremlin-driver` deve quindi essere configurato come segue nel codice Java:

```
import org.apache.tinkerpop.gremlin.driver.SigV4WebSocketChannelizer;
```

```
 ...

Cluster cluster = Cluster.build(your cluster)
                        .enableSsl(true)
                        .channelizer(SigV4WebSocketChannelizer.class)
                        .create();
Client client = cluster.connect();
client.submit("g.V().has('code','IAD')").all().get();
```

Connessione a Neptune tramite Java e SPARQL con firma di Signature Version 4 (RDF4J e Jena)

In questa sezione viene illustrato come connettersi a Neptune utilizzando RDF4J o Apache Jena con l'autenticazione di Signature Version 4.

Prerequisiti

- Java 8 o versione successiva.
- Apache Maven 3.3 o versione successiva.

  Per informazioni sull'installazione di questi prerequisiti su un'istanza EC2 che esegue Amazon Linux, consulta Prerequisiti su Amazon Linux EC2.
- Le credenziali di IAM per firmare le richieste. Per ulteriori informazioni, consulta Utilizzo della catena di provider delle credenziali predefinita nella Guida per gli sviluppatori di AWS SDK for Java

  > **Note**
  >
  > Se utilizzi credenziali temporanee, queste scadono dopo un intervallo specificato, comprensivo del token di sessione.
  > Devi aggiornare il token di sessione quando richiedi le nuove credenziali. Per ulteriori informazioni, consulta Using Temporary Security Credentials to Request Access to Resources nella IAM User Guide. AWS
- Imposta la variabile `SERVICE_REGION` su uno dei valori seguenti per indicare la regione dell'istanza database Neptune:
  - Stati Uniti orientali (Virginia settentrionale): `us-east-1`
  - Stati Uniti orientali (Ohio): `us-east-2`

- Stati Uniti occidentali (California settentrionale): `us-west-1`
- Stati Uniti occidentali (Oregon): `us-west-2`
- Canada (Centrale): `ca-central-1`
- Sud America (San Paolo): `sa-east-1`
- Europa (Stoccolma): `eu-north-1`
- Europa (Irlanda): `eu-west-1`
- Europa (Londra): `eu-west-2`
- Europa (Parigi): `eu-west-3`
- Europa (Francoforte): `eu-central-1`
- Medio Oriente (Bahrein): `me-south-1`
- Medio Oriente (Emirati Arabi Uniti): `me-central-1`
- Israele (Tel Aviv): `il-central-1`
- Africa (Città del Capo): `af-south-1`
- Asia Pacifico (Hong Kong): `ap-east-1`
- Asia Pacifico (Tokyo): `ap-northeast-1`
- Asia Pacifico (Seoul): `ap-northeast-2`
- Asia Pacifico (Osaka): `ap-northeast-3`
- Asia Pacifico (Singapore): `ap-southeast-1`
- Asia Pacifico (Sydney): `ap-southeast-2`
- Asia Pacifico (Mumbai): `ap-south-1`
- Cina (Pechino): `cn-north-1`
- Cina (Ningxia): `cn-northwest-1`
- AWS GovCloud (Stati Uniti occidentali): `us-gov-west-1`
- AWS GovCloud (Stati Uniti orientali): `us-gov-east-1`

Per connettersi a Neptune tramite RDF4J o Apache Jena con la firma di Signature Version 4

1. Clona il repository di esempio da. GitHub

```
git clone https://github.com/aws/amazon-neptune-sparql-java-sigv4.git
```

2. Passa alla directory clonata.

```
cd amazon-neptune-sparql-java-sigv4
```

3. Scarica la versione più recente del progetto controllando il ramo con il tag più recente.

```
git checkout $(git describe --tags `git rev-list --tags --max-count=1`)
```

4. Inserisci uno dei seguenti comandi per compilare ed eseguire il codice di esempio.

   Sostituisci *your-neptune-endpoint* con il nome host o l'indirizzo IP dell'istanza DB di Neptune. La porta predefinita è 8182.

   > **Note**
   >
   > Per informazioni su come trovare il nome host dell'istanza database Neptune, consulta la sezione Connessione agli endpoint Amazon Neptune.

   Eclipse RDF4J

   Immetti quanto segue per eseguire l'esempio di RDF4J.

```
mvn compile exec:java \
    -Dexec.mainClass="com.amazonaws.neptune.client.rdf4j.NeptuneRdf4JSigV4Example"
 \
    -Dexec.args="https://your-neptune-endpoint:port"
```

   Apache Jena

   Immetti quanto segue per eseguire l'esempio di Apache Jena.

```
mvn compile exec:java \
    -Dexec.mainClass="com.amazonaws.neptune.client.jena.NeptuneJenaSigV4Example" \
    -Dexec.args="https://your-neptune-endpoint:port"
```

5. Per visualizzare il codice sorgente dell'esempio, consulta gli esempi disponibili nella directory `src/main/java/com/amazonaws/neptune/client/`.

Per utilizzare il driver di firma SigV4 nella tua applicazione Java, aggiungi il pacchetto Maven `amazon-neptune-sigv4-signer` alla sezione `<dependencies>` di `pom.xml`. Ti consigliamo di usare gli esempi come punto di partenza.

Connessione a Neptune tramite SPARQL e Node.js con firma di Signature Version 4

Esecuzione di query utilizzando la firma Signature V4 e l'SDK per Javascript V3 AWS

Ecco un esempio di come connettersi a Neptune SPARQL utilizzando Node.js con l'autenticazione Signature Version 4 e AWS l'SDK per Javascript V3:

```
const { HttpRequest }  = require('@smithy/protocol-http');
const { fromNodeProviderChain } = require('@aws-sdk/credential-providers');
const { SignatureV4 } = require('@smithy/signature-v4');
const { Sha256 } = require('@aws-crypto/sha256-universal');
const https = require('https');

var region = 'us-west-2'; // e.g. us-west-1
var neptune_endpoint = 'your-Neptune-cluster-endpoint';  // like: 'cluster-
id.region.neptune.amazonaws.com'
var query = `query=PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX class: <http://aws.amazon.com/neptune/csv2rdf/class/>
PREFIX resource: <http://aws.amazon.com/neptune/csv2rdf/resource/>
PREFIX prop: <http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/>
PREFIX objprop: <http://aws.amazon.com/neptune/csv2rdf/objectProperty/>

SELECT ?movies ?title WHERE {
  ?jel prop:name "James Earl Jones" .
  ?movies ?p2 ?jel .
  ?movies prop:title ?title
} LIMIT 10`;

runQuery(query);

function runQuery(q) {
  var request = new HttpRequest({
    hostname: neptune_endpoint,
    port: 8182,
    path: 'sparql',
    body: encodeURI(query),
    headers: {
```

```
      'Content-Type': 'application/x-www-form-urlencoded',
      'host': neptune_endpoint + ':8182',
    },
    method: 'POST',
  });

  const credentialProvider = fromNodeProviderChain();
  let credentials = credentialProvider();
  credentials.then(
    (cred)=>{
      var signer = new SignatureV4({credentials: cred, region: region, sha256: Sha256,
 service: 'neptune-db'});
      signer.sign(request).then(
        (req)=>{
          var responseBody = '';
          var sendreq = https.request(
            {
              host: req.hostname,
              port: req.port,
              path: req.path,
              method: req.method,
              headers: req.headers,
            },
          (res) => {
            res.on('data', (chunk) => { responseBody += chunk; });
            res.on('end', () => {
                console.log(JSON.parse(responseBody));
            });
          });
          sendreq.write(req.body);
          sendreq.end();
        }
      );
    },
    (err)=>{
      console.error(err);
    }
  );
}
```

Esecuzione di query utilizzando la firma Signature V4 e l'SDK per Javascript V2 AWS

Ecco un esempio di come connettersi a Neptune SPARQL utilizzando Node.js con l'autenticazione Signature Version 4 e AWS l'SDK per Javascript V2:

```
var AWS = require('aws-sdk');

var region = 'us-west-2'; // e.g. us-west-1
var neptune_endpoint = 'your-Neptune-cluster-endpoint';  // like: 'cluster-
id.region.neptune.amazonaws.com'
var query = `query=PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX class: <http://aws.amazon.com/neptune/csv2rdf/class/>
PREFIX resource: <http://aws.amazon.com/neptune/csv2rdf/resource/>
PREFIX prop: <http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/>
PREFIX objprop: <http://aws.amazon.com/neptune/csv2rdf/objectProperty/>

SELECT ?movies ?title WHERE {
    ?jel prop:name "James Earl Jones" .
    ?movies ?p2 ?jel .
    ?movies prop:title ?title
} LIMIT 10`;

runQuery(query);

function runQuery(q) {

    var endpoint = new AWS.Endpoint(neptune_endpoint);
    endpoint.port = 8182;
    var request = new AWS.HttpRequest(endpoint, region);
    request.path += 'sparql';
    request.body = encodeURI(query);
    request.headers['Content-Type'] = 'application/x-www-form-urlencoded';
    request.headers['host'] = neptune_endpoint;
    request.method = 'POST';

    var credentials = new AWS.CredentialProviderChain();
    credentials.resolve((err, cred)=>{
        var signer = new AWS.Signers.V4(request, 'neptune-db');
        signer.addAuthorization(cred, new Date());
    });

    var client = new AWS.HttpClient();
    client.handleRequest(request, null, function(response) {
```

```
        console.log(response.statusCode + ' ' + response.statusMessage);
        var responseBody = '';
        response.on('data', function (chunk) {
            responseBody += chunk;
        });
        response.on('end', function (chunk) {
            console.log('Response body: ' + responseBody);
        });
    }, function(error) {
        console.log('Error: ' + error);
    });
}
```

Esempio: Connessione a Neptune tramite Python con la firma di Signature Version 4

In questa sezione viene mostrato un programma di esempio scritto in Python che spiega come lavorare con Signature Version 4 per Amazon Neptune. Questo esempio si basa sugli esempi nella sezione processo di firma Signature Version 4 in Riferimenti generali di Amazon Web Services.

Per lavorare con questo programma di esempio, è necessario:

- Python 3.x installato nel computer, scaricabile dal sito Python. Questi programmi sono stati testati utilizzando Python 3.6.
- La libreria di richieste Python, utilizzata nello script di esempio per creare richieste Web. Un modo comodo per installare i pacchetti Python è utilizzare `pip`, che consente di ottenere i pacchetti dal sito dell'indice dei pacchetti Python. È quindi possibile installare `requests` eseguendo `pip install requests` alla riga di comando.
- Una chiave di accesso (ID della chiave di accesso e chiave di accesso segreta) nelle variabili di ambiente denominate `AWS_ACCESS_KEY_ID` e `AWS_SECRET_ACCESS_KEY`. Come best practice, è consigliabile non incorporare le credenziali nel codice. Per ulteriori informazioni, consulta Best practice per gli account AWS nella Guida di riferimento AWS Account Management .

  La regione del tuo cluster database Neptune nella variabile di ambiente denominata `SERVICE_REGION`.

  Se usi credenziali temporanee, devi specificare `AWS_SESSION_TOKEN` oltre a `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` e `SERVICE_REGION`.

> **Note**
>
> Se utilizzi credenziali temporanee, queste scadono dopo un intervallo specificato, comprensivo del token di sessione.
> Devi aggiornare il token di sessione quando richiedi le nuove credenziali. Per ulteriori informazioni, consulta Utilizzo delle credenziali di sicurezza temporanee per richiedere l'accesso alle risorse AWS.

L'esempio seguente mostra come effettuare richieste firmate a Neptune utilizzando Python. La richiesta effettua una richiesta GET o POST. Le informazioni di autenticazione vengono trasferite utilizzando l'intestazione della richiesta `Authorization`.

Questo esempio funziona anche come funzione. AWS Lambda Per ulteriori informazioni, consulta the section called “Impostazione di Lambda”.

Per effettuare richieste firmate agli endpoint Neptune Gremlin e SPARQL

1. Crea un nuovo file denominato `neptunesigv4.py` e aprilo in un editor di testo.
2. Copia la seguente chiave pubblica e incollala nel file `neptunesigv4.py`.

```
# Amazon Neptune version 4 signing example (version v3)

# The following script requires python 3.6+
#    (sudo yum install python36 python36-virtualenv python36-pip)
# => the reason is that we're using urllib.parse() to manually encode URL
#    parameters: the problem here is that SIGV4 encoding requires whitespaces
#    to be encoded as %20 rather than not or using '+', as done by previous/
#    default versions of the library.


# See: https://docs.aws.amazon.com/general/latest/gr/sigv4_signing.html
import sys, datetime, hashlib, hmac
import requests  # pip3 install requests
import urllib
import os
import json
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import ReadOnlyCredentials
```

```
from types import SimpleNamespace
from argparse import RawTextHelpFormatter
from argparse import ArgumentParser

# Configuration. https is required.
protocol = 'https'

# The following lines enable debugging at httplib level (requests->urllib3-
>http.client)
# You will see the REQUEST, including HEADERS and DATA, and RESPONSE with HEADERS
 but without DATA.
#
# The only thing missing will be the response.body which is not logged.
#
# import logging
# from http.client import HTTPConnection
# HTTPConnection.debuglevel = 1
# logging.basicConfig()
# logging.getLogger().setLevel(logging.DEBUG)
# requests_log = logging.getLogger("requests.packages.urllib3")
# requests_log.setLevel(logging.DEBUG)
# requests_log.propagate = True


# Read AWS access key from env. variables. Best practice is NOT
# to embed credentials in code.
access_key = os.getenv('AWS_ACCESS_KEY_ID', '')
secret_key = os.getenv('AWS_SECRET_ACCESS_KEY', '')
region = os.getenv('SERVICE_REGION', '')

# AWS_SESSION_TOKEN is optional environment variable. Specify a session token only
 if you are using temporary
# security credentials.
session_token = os.getenv('AWS_SESSION_TOKEN', '')

### Note same script can be used for AWS Lambda (runtime = python3.6).
## Steps to use this python script for AWS Lambda
# 1. AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY and AWS_SESSION_TOKEN and AWS_REGION
 variables are already part of Lambda's Execution environment
#    No need to set them up explicitly.
# 3. Create Lambda deployment package https://docs.aws.amazon.com/lambda/latest/dg/
lambda-python-how-to-create-deployment-package.html
# 4. Create a Lambda function in the same VPC and assign an IAM role with neptune
 access
```

```
def lambda_handler(event, context):
    # sample_test_input = {
    #     "host": "END_POINT:8182",
    #     "method": "GET",
    #     "query_type": "gremlin",
    #     "query": "g.V().count()"
    # }

    # Lambda uses AWS_REGION instead of SERVICE_REGION
    global region
    region = os.getenv('AWS_REGION', '')

    host = event['host']
    method = event['method']
    query_type = event['query_type']
    query =  event['query']

    return make_signed_request(host, method, query_type, query)

def validate_input(method, query_type):
    # Supporting GET and POST for now:
    if (method != 'GET' and method != 'POST'):
        print('First parameter must be "GET" or "POST", but is "' + method + '".')
        sys.exit()

    # SPARQL UPDATE requires POST
    if (method == 'GET' and query_type == 'sparqlupdate'):
        print('SPARQL UPDATE is not supported in GET mode. Please choose POST.')
        sys.exit()

def get_canonical_uri_and_payload(query_type, query, method):
    # Set the stack and payload depending on query_type.
    if (query_type == 'sparql'):
        canonical_uri = '/sparql/'
        payload = {'query': query}

    elif (query_type == 'sparqlupdate'):
        canonical_uri = '/sparql/'
        payload = {'update': query}

    elif (query_type == 'gremlin'):
        canonical_uri = '/gremlin/'
        payload = {'gremlin': query}
```

```
        if (method == 'POST'):
            payload = json.dumps(payload)

    elif (query_type == 'openCypher'):
        canonical_uri = '/openCypher/'
        payload = {'query': query}

    elif (query_type == "loader"):
        canonical_uri = "/loader/"
        payload = query

    elif (query_type == "status"):
        canonical_uri = "/status/"
        payload = {}

    elif (query_type == "gremlin/status"):
        canonical_uri = "/gremlin/status/"
        payload = {}

    elif (query_type == "openCypher/status"):
        canonical_uri = "/openCypher/status/"
        payload = {}

    elif (query_type == "sparql/status"):
        canonical_uri = "/sparql/status/"
        payload = {}

    else:
        print(
            'Third parameter should be from ["gremlin", "sparql", "sparqlupdate",
 "loader", "status] but is "' + query_type + '".')
        sys.exit()
    ## return output as tuple
    return canonical_uri, payload

def make_signed_request(host, method, query_type, query):
    service = 'neptune-db'
    endpoint = protocol + '://' + host

    print()
    print('+++++ USER INPUT +++++')
    print('host = ' + host)
    print('method = ' + method)
    print('query_type = ' + query_type)
```

```
   print('query = ' + query)

   # validate input
   validate_input(method, query_type)

   # get canonical_uri and payload
   canonical_uri, payload = get_canonical_uri_and_payload(query_type, query,
method)

   # assign payload to data or params
   data = payload if method == 'POST' else None
   params = payload if method == 'GET' else None

   # create request URL
   request_url = endpoint + canonical_uri

   # create and sign request
   creds = SimpleNamespace(
       access_key=access_key, secret_key=secret_key, token=session_token,
region=region,
   )

   request = AWSRequest(method=method, url=request_url, data=data, params=params)
   SigV4Auth(creds, service, region).add_auth(request)

   r = None

   # ************* SEND THE REQUEST *************
   if (method == 'GET'):

       print('++++ BEGIN GET REQUEST +++++')
       print('Request URL = ' + request_url)
       r = requests.get(request_url, headers=request.headers, verify=False,
params=params)

   elif (method == 'POST'):

       print('\n+++++ BEGIN POST REQUEST +++++')
       print('Request URL = ' + request_url)
       if (query_type == "loader"):
           request.headers['Content-type'] = 'application/json'
       r = requests.post(request_url, headers=request.headers, verify=False,
data=data)
```

```
    else:
        print('Request method is neither "GET" nor "POST", something is wrong
 here.')

    if r is not None:
        print()
        print('+++++ RESPONSE +++++')
        print('Response code: %d\n' % r.status_code)
        response = r.text
        r.close()
        print(response)

        return response

help_msg = '''
    export AWS_ACCESS_KEY_ID=[MY_ACCESS_KEY_ID]
    export AWS_SECRET_ACCESS_KEY=[MY_SECRET_ACCESS_KEY]
    export AWS_SESSION_TOKEN=[MY_AWS_SESSION_TOKEN]
    export SERVICE_REGION=[us-east-1|us-east-2|us-west-2|eu-west-1]

    python version >=3.6 is required.

    Examples: For help
    python3 program_name.py -h

    Examples: Queries
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q status
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql/
status
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql -d
 "SELECT ?s WHERE { ?s ?p ?o }"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q sparql -d
 "SELECT ?s WHERE { ?s ?p ?o }"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q
 sparqlupdate -d "INSERT DATA { <https://s> <https://p> <https://o> }"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin/
status
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin -d
 "g.V().count()"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q gremlin -d
 "g.V().count()"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher/
status
```

```
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher
 -d "MATCH (n1) RETURN n1 LIMIT 1;"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q openCypher
 -d "MATCH (n1) RETURN n1 LIMIT 1;"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{"loadId": "68b28dcc-8e15-02b1-133d-9bd0557607e6"}'
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{}'
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q loader
 -d '{"source": "source", "format" : "csv", "failOnError": "fail_on_error",
 "iamRoleArn": "iam_role_arn", "region": "region"}'

    Environment variables must be defined as AWS_ACCESS_KEY_ID,
 AWS_SECRET_ACCESS_KEY and SERVICE_REGION.
    You should also set AWS_SESSION_TOKEN environment variable if you are using
 temporary credentials (ex. IAM Role or EC2 Instance profile).

    Current Limitations:
    - Query mode "sparqlupdate" requires POST (as per the SPARQL 1.1 protocol)
            '''

def exit_and_print_help():
    print(help_msg)
    exit()

def parse_input_and_query_neptune():


    parser = ArgumentParser(description=help_msg,
 formatter_class=RawTextHelpFormatter)
    group_host = parser.add_mutually_exclusive_group()
    group_host.add_argument("-ho", "--host", type=str)
    group_port = parser.add_mutually_exclusive_group()
    group_port.add_argument("-p", "--port", type=int, help="port ex. 8182,
 default=8182", default=8182)
    group_action = parser.add_mutually_exclusive_group()
    group_action.add_argument("-a", "--action", type=str, help="http action,
 default = GET", default="GET")
    group_endpoint = parser.add_mutually_exclusive_group()
    group_endpoint.add_argument("-q", "--query_type", type=str, help="query_type,
 default = status ", default="status")
    group_data = parser.add_mutually_exclusive_group()
    group_data.add_argument("-d", "--data", type=str, help="data required for the
 http action", default="")
```

```
    args = parser.parse_args()
    print(args)

    # Read command line parameters
    host = args.host
    port = args.port
    method = args.action
    query_type = args.query_type
    query = args.data

    if (access_key == ''):
        print('!!! ERROR: Your AWS_ACCESS_KEY_ID environment variable is
 undefined.')
        exit_and_print_help()

    if (secret_key == ''):
        print('!!! ERROR: Your AWS_SECRET_ACCESS_KEY environment variable is
 undefined.')
        exit_and_print_help()

    if (region == ''):
        print('!!! ERROR: Your SERVICE_REGION environment variable is undefined.')
        exit_and_print_help()

    if host is None:
        print('!!! ERROR: Neptune DNS is missing')
        exit_and_print_help()

    host = host + ":" + str(port)
    make_signed_request(host, method, query_type, query)


if __name__ == "__main__":
    parse_input_and_query_neptune()
```

3. In un terminale, individua la posizione del file `neptunesigv4.py`.
4. Inserisci i seguenti comandi, sostituendo la chiave di accesso, la chiave segreta e la regione con i valori corretti.

```
export AWS_ACCESS_KEY_ID=MY_ACCESS_KEY_ID
export AWS_SECRET_ACCESS_KEY=MY_SECRET_ACCESS_KEY
```

```
export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or eu-
west-3 or eu-central-1 or me-south-1 or
                      me-central-1 or il-central-1 or af-south-1 or ap-east-1 or
 ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-southeast-2 or ap-south-1
 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

Se usi credenziali temporanee, devi specificare `AWS_SESSION_TOKEN` oltre a `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` e `SERVICE_REGION`.

```
export AWS_SESSION_TOKEN=MY_AWS_SESSION_TOKEN
```

> **Note**
>
> Se utilizzi credenziali temporanee, queste scadono dopo un intervallo specificato, comprensivo del token di sessione.
> Devi aggiornare il token di sessione quando richiedi le nuove credenziali. Per ulteriori informazioni, consulta Utilizzo delle credenziali di sicurezza temporanee per richiedere l'accesso alle risorse AWS.

5. Inserisci uno dei seguenti comandi per inviare una richiesta firmata all'istanza database Neptune. Questi esempi utilizzano Python versione 3.6.

   Stato endpoint

   ```
   python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q status
   ```

   Gremlin

   ```
   python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin -d
    "g.V().count()"

   python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q gremlin -d
    "g.V().count()"
   ```

   Stato Gremlin

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin/
status
```

SPARQL

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql -d
 "SELECT ?s WHERE { ?s ?p ?o }"
```

SPARQL UPDATE

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q sparqlupdate
 -d "INSERT DATA { <https://s> <https://p> <https://o> }"
```

Stato SPARQL

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql/status
```

openCypher

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher -d
 "MATCH (n1) RETURN n1 LIMIT 1;"

python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q openCypher -
d "MATCH (n1) RETURN n1 LIMIT 1;"
```

Stato openCypher

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher/
status
```

Loader

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{"loadId": "68b28dcc-8e15-02b1-133d-9bd0557607e6"}'

python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{}'
```

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q loader
 -d '{"source": "source", "format" : "csv", "failOnError": "fail_on_error",
 "iamRoleArn": "iam_role_arn", "region": "region"}'
```

6. La sintassi per eseguire lo script Python è la seguente:

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p port -a GET|POST -q gremlin|
sparql|sparqlupdate|loader|status -d "string0data"
```

SPARQL UPDATE richiede POST.

Gestione dell'accesso con le policy IAM

Le policy IAM sono oggetti JSON che definiscono le autorizzazioni per utilizzare azioni e risorse.

È possibile controllare l'accesso AWS creando policy e associandole a AWS identità o risorse. Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un principale (utente, utente root o sessione di ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente di IAM.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. Successivamente l'amministratore può aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'azione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'azione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dall' AWS Management Console AWS CLI, dall'o dall' AWS API.

Policy basate sulle identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruoli IAM). Tali policy definiscono le azioni che utenti e ruoli

possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più utenti, gruppi e ruoli nel tuo Account AWS. Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente di IAM.

Utilizzo delle politiche di controllo dei servizi (SCP) con le organizzazioni AWS

Le policy di controllo dei servizi (SCP) sono politiche JSON che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in. AWS Organizations AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più AWS account di proprietà dell'azienda. Se abiliti tutte le funzionalità in un'organizzazione, puoi applicare le policy di controllo dei servizi (SCP) a uno o tutti i tuoi account. L'SCP limita le autorizzazioni per le entità negli account dei membri, incluso ogni AWS utente root dell'account. Per ulteriori informazioni su Organizations and SCP, consulta How SCP work in the AWS Organizations User Guide.

I clienti che implementano Amazon Neptune AWS in un account AWS all'interno di un'organizzazione possono sfruttare gli SCP per controllare quali account possono utilizzare Neptune. Per garantire l'accesso a Neptune all'interno di un account membro, assicurarsi di consentire l'accesso alle azioni IAM del piano di controllo (control-plane) e del piano dati utilizzando rispettivamente `neptune:*` e `neptune-db:*`.

Autorizzazioni necessarie per utilizzare la console Amazon Neptune

Affinché un utente possa utilizzare la console Amazon Neptune, è necessario che disponga di un set minimo di autorizzazioni. Queste autorizzazioni permettono all'utente di descrivere le risorse Neptune per il proprio account AWS e di fornire altre informazioni correlate, tra cui informazioni sulla sicurezza e sulla rete di Amazon EC2.

Se decidi di creare una policy IAM più restrittiva delle autorizzazioni minime richieste, la console non funzionerà come previsto per gli utenti con tale policy IAM. Per garantire che gli utenti possano continuare a utilizzare la console Neptune, collega anche la policy gestita `NeptuneReadOnlyAccess` all'utente, come descritto in AWS politiche gestite (predefinite) per Amazon Neptune.

Non è necessario consentire autorizzazioni minime per la console per gli utenti che effettuano chiamate solo verso l' AWS CLI API Amazon Neptune.

Collegamento di una policy IAM a un utente IAM

Per applicare una policy gestita o personalizzata, collegarla a un utente IAM. Per un tutorial su questo argomento, consulta Creare e collegare la tua prima policy gestita dal cliente nella Guida per l'utente IAM.

Durante il tutorial, puoi utilizzare uno degli esempi di policy indicati in questa sezione come punto di partenza e personalizzarli in base alle tue esigenze. Al termine del tutorial, avrai un utente IAM con una policy collegata che potrà utilizzare l'operazione `neptune-db:*`.

> **Important**
>
> - Le modifiche apportate a una policy IAM richiedono fino a 10 minuti per essere applicate alle risorse Neptune specificate.
> - Le policy IAM applicate a un cluster database Neptune si applicano a tutte le istanze del cluster.

Utilizzo di diversi tipi di policy IAM per controllare l'accesso a Neptune

Per fornire l'accesso alle azioni amministrative di Neptune o ai dati in un cluster database Neptune, collegare le policy a un utente o un ruolo IAM. Per informazioni sul collegamento di una policy IAM a un utente, consulta Collegamento di una policy IAM a un utente IAM. Per informazioni su come collegare una policy a un ruolo, consulta Aggiunta e rimozione delle policy IAM nella Guida per l'utente di IAM.

Per l'accesso generale a Neptune, puoi utilizzare una delle policy gestite di Neptune. Per un accesso più limitato, puoi creare la tua policy personalizzata utilizzando le azioni e le risorse amministrative supportate da Neptune.

In una policy IAM personalizzata, puoi utilizzare due diversi tipi di dichiarazioni di policy che controllano diverse modalità di accesso a un cluster database Neptune:

- Dichiarazioni di policy amministrative: le dichiarazioni di policy amministrative forniscono l'accesso alle API di gestione Neptune utilizzate per creare, configurare e gestire un cluster database e le relative istanze.

Poiché Neptune condivide funzionalità con Amazon RDS, le azioni amministrative, le risorse e le chiavi di condizione nelle policy Neptune utilizzano un prefisso `rds:` per impostazione predefinita.

- Dichiarazioni di policy di accesso ai dati: le dichiarazioni di policy di accesso ai dati utilizzano azioni di accesso ai dati, risorse e chiavi di condizione per controllare l'accesso ai dati contenuti in un cluster database.

  Le azioni di accesso ai dati, le risorse e le chiavi di condizione di Neptune utilizzano un prefisso `neptune-db:`.

Utilizzo delle chiavi di contesto delle condizioni IAM in Amazon Neptune

È possibile specificare le condizioni in una dichiarazione di policy IAM che controlla l'accesso a Neptune. La dichiarazione di policy diventa effettiva solo quando le condizioni sono true.

Ad esempio, potresti volere che una dichiarazione di policy diventi effettiva solo dopo una data specifica o che consenta l'accesso solo quando nella richiesta è presente un valore specifico.

Per specificare le condizioni, è possibile utilizzare le chiavi di condizione predefinite nell'elemento `Condition` di una dichiarazione di policy insieme agli operatori della policy della condizione IAM come uguale a o minore di.

Se specifichi più elementi `Condition` in un'istruzione o più chiavi in un singolo elemento `Condition`, questi vengono valutati da AWS utilizzando un'operazione `AND` logica. Se specifichi più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'operazione logica. `OR` Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

Puoi anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, puoi autorizzare un utente IAM ad accedere a una risorsa solo se è stata taggata con il relativo nome utente IAM. Per ulteriori informazioni, consulta Elementi delle policy IAM: variabili e tag nella Guida per l'utente di IAM.

Il tipo di dati di una chiave di condizione determina quali operatori di condizione è possibile utilizzare per confrontare i valori nella richiesta con i valori della dichiarazione di policy. Se si utilizza un operatore di condizione che non è compatibile con tale tipo di dati, la corrispondenza ha sempre esito negativo e la dichiarazione di policy non viene mai applicata.

Neptune supporta diversi set di chiavi di condizione per le dichiarazioni di policy amministrative rispetto alle dichiarazioni di policy di accesso ai dati:

- Chiavi di condizione per le dichiarazioni di policy amministrative
- Chiavi di condizione per le dichiarazioni di policy di accesso ai dati

Supporto delle funzionalità delle policy IAM e di controllo degli accessi in Amazon Neptune

La tabella seguente mostra le funzionalità IAM supportate da Neptune per le dichiarazioni di policy amministrative e le dichiarazioni di policy di accesso ai dati:

Funzionalità IAM che è possibile utilizzare con Neptune

Funzionalità IAM	Amministrativa	Accesso ai dati
Policy basate su identità	Sì	Sì
Policy basate su risorse	No	No
Azioni di policy	Sì	Sì
Risorse relative alle policy	Sì	Sì
Chiavi della condizione globale	Sì	(un sottoinsieme)
Chiavi di condizione basate su tag	Sì	No
Liste di controllo degli accessi (ACL)	No	No
Policy di controllo dei servizi (Service Control Policies, SCP)	Sì	Sì
Ruoli collegati ai servizi	Sì	No

Limitazioni della policy IAM

Le modifiche apportate a una policy IAM richiedono fino a 10 minuti per essere applicate alle risorse Neptune specificate.

Le policy IAM applicate a un cluster database Neptune si applicano a tutte le istanze del cluster.

Neptune attualmente non supporta il controllo dell'accesso multi-account.

AWS politiche gestite (predefinite) per Amazon Neptune

AWS affronta molti casi d'uso comuni fornendo policy IAM autonome create e amministrate da. AWS Le policy gestite concedono le autorizzazioni necessarie per i casi di utilizzo comune in modo da non dover cercare quali sono le autorizzazioni richieste. Per ulteriori informazioni, consulta Policy gestite da AWS nella Guida per l'utente di IAM.

Le seguenti policy AWS gestite, che puoi allegare agli utenti del tuo account, servono per l'utilizzo delle API di gestione di Amazon Neptune:

- NeptuneReadOnlyAccess— Garantisce l'accesso in sola lettura a tutte le risorse di Neptune per scopi amministrativi e di accesso ai dati nell'account root. AWS
- NeptuneFullAccess— Garantisce l'accesso completo a tutte le risorse di Neptune per scopi amministrativi e di accesso ai dati nell'account root. AWS Questa opzione è consigliata se è necessario l'accesso completo a Neptune dall'SDK o, ma non per AWS CLI l'accesso. AWS Management Console
- NeptuneConsoleFullAccess— Concede l'accesso completo nell' AWS account root a tutte le azioni e risorse amministrative di Neptune, ma non a tutte le azioni o risorse di accesso ai dati. Include inoltre autorizzazioni aggiuntive per semplificare l'accesso a Neptune dalla console, comprese autorizzazioni IAM e Amazon EC2 (VPC) limitate.
- NeptuneGraphReadOnlyAccess — Fornisce accesso in sola lettura a tutte le risorse di Amazon Neptune Analytics insieme a autorizzazioni di sola lettura per i servizi dipendenti
- AWSServiceRoleForNeptuneGraphPolicy— Consente ai grafici di Neptune Analytics di CloudWatch pubblicare metriche e registri operativi e di utilizzo.

Le policy e i ruoli IAM Neptune concedono l'accesso alle risorse Amazon RDS, perché Neptune condivide tecnologia operativa con Amazon RDS per determinate funzionalità di gestione. Ciò include le autorizzazioni API amministrative, motivo per cui le azioni amministrative di Neptune hanno un prefisso `rds:`.

Aggiornamenti alle politiche gestite da AWS Neptune

La tabella seguente tiene traccia degli aggiornamenti alle policy gestite da Neptune a partire dal momento in cui Neptune ha iniziato a tenere traccia di queste modifiche:

Policy	Descrizione	Data
AWS politiche gestite per Amazon Neptune: aggiornamento delle politiche esistenti	Le `NeptuneReadOnlyAccess` politiche `NeptuneFullAccess` gestite ora includono `Sid` (statement ID) come identificatore nella dichiarazione politica.	22/01/2024
NeptuneGraphReadOnlyAccess(rilasciato)	Rilasciato per fornire accesso in sola lettura ai grafici e alle risorse di Analisi Neptune.	29/11/23
AWSServiceRoleForNeptuneGraphPolicy(rilasciato)	Rilasciato per consentire ai grafici di Neptune Analytics di accedere CloudWatch alla pubblicazione di metriche e registri operativi e di utilizzo. Consulta Using service-linked roles (SLRs) in Neptune Analytics.	2023-11-29
NeptuneConsoleFullAccess(autorizzazioni aggiunte)	Le autorizzazioni aggiunte forniscono l'accesso necessario per interagire con i grafici di Analisi Neptune.	29/11/2023
NeptuneFullAccess(autorizzazioni aggiunte)	Sono state aggiunte autorizzazioni di accesso ai dati e autorizzazioni per le nuove API del database globale.	28/07/2022

Policy	Descrizione	Data
NeptuneConsoleFullAccess(autorizzazioni aggiunte)	Sono state aggiunte autorizzazioni per le nuove API del database globale.	21-07-2022
Neptune ha iniziato a tenere traccia delle modifiche	Neptune ha iniziato a tenere traccia delle modifiche alle AWS sue politiche gestite.	2022-07-21

Policy AWS gestita da **`NeptuneReadOnlyAccess`**

La politica NeptuneReadOnlyAccessgestita di seguito garantisce l'accesso in sola lettura a tutte le azioni e le risorse di Neptune per scopi amministrativi e di accesso ai dati.

> **Note**
>
> Questa policy è stata aggiornata il 21 luglio 2022 per includere le autorizzazioni di accesso ai dati di sola lettura e le autorizzazioni amministrative di sola lettura e per includere le autorizzazioni per le azioni del database globale.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowReadOnlyPermissionsForRDS",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
```

```
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DownloadDBLogFilePortion",
                "rds:ListTagsForResource"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForCloudwatch",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForEC2",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForKMS",
            "Effect": "Allow",
            "Action": [
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "kms:ListAliases",
```

```
                "kms:ListKeyPolicies"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForLogs",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams",
                "logs:GetLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*",
                "arn:aws:logs:*:*:log-group:/aws/neptune/*:log-stream:*"
            ]
        },
        {
            "Sid": "AllowReadOnlyPermissionsForNeptuneDB",
            "Effect": "Allow",
            "Action": [
                "neptune-db:Read*",
                "neptune-db:Get*",
                "neptune-db:List*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Policy AWS gestita da **NeptuneFullAccess**

La politica NeptuneFullAccessgestita di seguito garantisce l'accesso completo a tutte le azioni e le risorse di Neptune per scopi amministrativi e di accesso ai dati. È consigliato se è necessario l'accesso completo da AWS CLI o da un SDK, ma non da. AWS Management Console

> **Note**
>
> Questa policy è stata aggiornata il 21 luglio 2022 per includere le autorizzazioni di accesso ai dati complete e le autorizzazioni amministrative complete e per includere le autorizzazioni per le azioni del database globale.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowNeptuneCreate",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBCluster",
                "rds:CreateDBInstance"
            ],
            "Resource": [
                "arn:aws:rds:*:*:*"
            ],
            "Condition": {
                "StringEquals": {
                    "rds:DatabaseEngine": [
                        "graphdb",
                        "neptune"
                    ]
                }
            }
        },
        {
            "Sid": "AllowManagementPermissionsForRDS",
            "Effect": "Allow",
            "Action": [
                "rds:AddRoleToDBCluster",
                "rds:AddSourceIdentifierToSubscription",
                "rds:AddTagsToResource",
                "rds:ApplyPendingMaintenanceAction",
                "rds:CopyDBClusterParameterGroup",
                "rds:CopyDBClusterSnapshot",
                "rds:CopyDBParameterGroup",
                "rds:CreateDBClusterEndpoint",
                "rds:CreateDBClusterParameterGroup",
                "rds:CreateDBClusterSnapshot",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSubnetGroup",
                "rds:CreateEventSubscription",
                "rds:CreateGlobalCluster",
                "rds:DeleteDBCluster",
                "rds:DeleteDBClusterEndpoint",
                "rds:DeleteDBClusterParameterGroup",
```

```
                "rds:DeleteDBClusterSnapshot",
                "rds:DeleteDBInstance",
                "rds:DeleteDBParameterGroup",
                "rds:DeleteDBSubnetGroup",
                "rds:DeleteEventSubscription",
                "rds:DeleteGlobalCluster",
                "rds:DescribeDBClusterEndpoints",
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEngineDefaultClusterParameters",
                "rds:DescribeEngineDefaultParameters",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOptionGroups",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DescribeValidDBInstanceModifications",
                "rds:DownloadDBLogFilePortion",
                "rds:FailoverDBCluster",
                "rds:FailoverGlobalCluster",
                "rds:ListTagsForResource",
                "rds:ModifyDBCluster",
                "rds:ModifyDBClusterEndpoint",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyDBSubnetGroup",
                "rds:ModifyEventSubscription",
                "rds:ModifyGlobalCluster",
```

```
                "rds:PromoteReadReplicaDBCluster",
                "rds:RebootDBInstance",
                "rds:RemoveFromGlobalCluster",
                "rds:RemoveRoleFromDBCluster",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:RemoveTagsFromResource",
                "rds:ResetDBClusterParameterGroup",
                "rds:ResetDBParameterGroup",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:RestoreDBClusterToPointInTime",
                "rds:StartDBCluster",
                "rds:StopDBCluster"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "AllowOtherDepedentPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs",
                "kms:ListAliases",
                "kms:ListKeyPolicies",
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "sns:ListSubscriptions",
                "sns:ListTopics",
                "sns:Publish"
            ],
            "Resource": [
                "*"
            ]
        },
        {
```

```
            "Sid": "AllowPassRoleForNeptune",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:passedToService": "rds.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowCreateSLRForNeptune",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "rds.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowDataAccessForNeptune",
            "Effect": "Allow",
            "Action": [
                "neptune-db:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Policy AWS gestita da **NeptuneConsoleFullAccess**

La politica NeptuneConsoleFullAccessgestita di seguito garantisce l'accesso completo a tutte le azioni e le risorse di Neptune per scopi amministrativi, ma non per scopi di accesso ai dati. Include inoltre autorizzazioni aggiuntive per semplificare l'accesso a Neptune dalla console, comprese autorizzazioni IAM e Amazon EC2 (VPC) limitate.

> **Note**
>
> Questa policy è stata aggiornata il 29 novembre 2023 per includere le autorizzazioni necessarie per interagire con i grafici di Analisi Neptune.
> È stata aggiornata il 21 luglio 2022 per includere le autorizzazioni per le azioni del database globale.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowNeptuneCreate",
      "Effect": "Allow",
      "Action": [
        "rds:CreateDBCluster",
        "rds:CreateDBInstance"
      ],
      "Resource": [
        "arn:aws:rds:*:*:*"
      ],
      "Condition": {
        "StringEquals": {
          "rds:DatabaseEngine": [
            "graphdb",
            "neptune"
          ]
        }
      }
    },
    {
      "Sid": "AllowManagementPermissionsForRDS",
      "Action": [
        "rds:AddRoleToDBCluster",
        "rds:AddSourceIdentifierToSubscription",
        "rds:AddTagsToResource",
        "rds:ApplyPendingMaintenanceAction",
        "rds:CopyDBClusterParameterGroup",
        "rds:CopyDBClusterSnapshot",
        "rds:CopyDBParameterGroup",
        "rds:CreateDBClusterParameterGroup",
        "rds:CreateDBClusterSnapshot",
```

```
        "rds:CreateDBParameterGroup",
        "rds:CreateDBSubnetGroup",
        "rds:CreateEventSubscription",
        "rds:DeleteDBCluster",
        "rds:DeleteDBClusterParameterGroup",
        "rds:DeleteDBClusterSnapshot",
        "rds:DeleteDBInstance",
        "rds:DeleteDBParameterGroup",
        "rds:DeleteDBSubnetGroup",
        "rds:DeleteEventSubscription",
        "rds:DescribeAccountAttributes",
        "rds:DescribeCertificates",
        "rds:DescribeDBClusterParameterGroups",
        "rds:DescribeDBClusterParameters",
        "rds:DescribeDBClusterSnapshotAttributes",
        "rds:DescribeDBClusterSnapshots",
        "rds:DescribeDBClusters",
        "rds:DescribeDBEngineVersions",
        "rds:DescribeDBInstances",
        "rds:DescribeDBLogFiles",
        "rds:DescribeDBParameterGroups",
        "rds:DescribeDBParameters",
        "rds:DescribeDBSecurityGroups",
        "rds:DescribeDBSubnetGroups",
        "rds:DescribeEngineDefaultClusterParameters",
        "rds:DescribeEngineDefaultParameters",
        "rds:DescribeEventCategories",
        "rds:DescribeEventSubscriptions",
        "rds:DescribeEvents",
        "rds:DescribeOptionGroups",
        "rds:DescribeOrderableDBInstanceOptions",
        "rds:DescribePendingMaintenanceActions",
        "rds:DescribeValidDBInstanceModifications",
        "rds:DownloadDBLogFilePortion",
        "rds:FailoverDBCluster",
        "rds:ListTagsForResource",
        "rds:ModifyDBCluster",
        "rds:ModifyDBClusterParameterGroup",
        "rds:ModifyDBClusterSnapshotAttribute",
        "rds:ModifyDBInstance",
        "rds:ModifyDBParameterGroup",
        "rds:ModifyDBSubnetGroup",
        "rds:ModifyEventSubscription",
        "rds:PromoteReadReplicaDBCluster",
```

```
        "rds:RebootDBInstance",
        "rds:RemoveRoleFromDBCluster",
        "rds:RemoveSourceIdentifierFromSubscription",
        "rds:RemoveTagsFromResource",
        "rds:ResetDBClusterParameterGroup",
        "rds:ResetDBParameterGroup",
        "rds:RestoreDBClusterFromSnapshot",
        "rds:RestoreDBClusterToPointInTime"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowOtherDepedentPermissions",
      "Action": [
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:ListMetrics",
        "ec2:AllocateAddress",
        "ec2:AssignIpv6Addresses",
        "ec2:AssignPrivateIpAddresses",
        "ec2:AssociateAddress",
        "ec2:AssociateRouteTable",
        "ec2:AssociateSubnetCidrBlock",
        "ec2:AssociateVpcCidrBlock",
        "ec2:AttachInternetGateway",
        "ec2:AttachNetworkInterface",
        "ec2:CreateCustomerGateway",
        "ec2:CreateDefaultSubnet",
        "ec2:CreateDefaultVpc",
        "ec2:CreateInternetGateway",
        "ec2:CreateNatGateway",
        "ec2:CreateNetworkInterface",
        "ec2:CreateRoute",
        "ec2:CreateRouteTable",
        "ec2:CreateSecurityGroup",
        "ec2:CreateSubnet",
        "ec2:CreateVpc",
        "ec2:CreateVpcEndpoint",
        "ec2:CreateVpcEndpoint",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAddresses",
```

```
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeCustomerGateways",
        "ec2:DescribeInstances",
        "ec2:DescribeNatGateways",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribePrefixLists",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroupReferences",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs",
        "ec2:DescribeVpcs",
        "ec2:ModifyNetworkInterfaceAttribute",
        "ec2:ModifySubnetAttribute",
        "ec2:ModifyVpcAttribute",
        "ec2:ModifyVpcEndpoint",
        "iam:ListRoles",
        "kms:ListAliases",
        "kms:ListKeyPolicies",
        "kms:ListKeys",
        "kms:ListRetirableGrants",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents",
        "sns:ListSubscriptions",
        "sns:ListTopics",
        "sns:Publish"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowPassRoleForNeptune",
      "Action": "iam:PassRole",
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
```

```
        "StringEquals": {
          "iam:passedToService": "rds.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowCreateSLRForNeptune",
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "rds.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowManagementPermissionsForNeptuneAnalytics",
      "Effect": "Allow",
      "Action": [
        "neptune-graph:CreateGraph",
        "neptune-graph:DeleteGraph",
        "neptune-graph:GetGraph",
        "neptune-graph:ListGraphs",
        "neptune-graph:UpdateGraph",
        "neptune-graph:ResetGraph",
        "neptune-graph:CreateGraphSnapshot",
        "neptune-graph:DeleteGraphSnapshot",
        "neptune-graph:GetGraphSnapshot",
        "neptune-graph:ListGraphSnapshots",
        "neptune-graph:RestoreGraphFromSnapshot",
        "neptune-graph:CreatePrivateGraphEndpoint",
        "neptune-graph:GetPrivateGraphEndpoint",
        "neptune-graph:ListPrivateGraphEndpoints",
        "neptune-graph:DeletePrivateGraphEndpoint",
        "neptune-graph:CreateGraphUsingImportTask",
        "neptune-graph:GetImportTask",
        "neptune-graph:ListImportTasks",
        "neptune-graph:CancelImportTask"
      ],
      "Resource": [
        "arn:aws:neptune-graph:*:*:*"
      ]
```

```
    },
    {
      "Sid": "AllowPassRoleForNeptuneAnalytics",
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:passedToService": "neptune-graph.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowCreateSLRForNeptuneAnalytics",
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::*:role/aws-service-role/neptune-graph.amazonaws.com/
AWSServiceRoleForNeptuneGraph",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "neptune-graph.amazonaws.com"
        }
      }
    }
  ]
}
```

Policy AWS gestita da **NeptuneGraphReadOnlyAccess**

La politica NeptuneGraphReadOnlyAccessgestita riportata di seguito fornisce l'accesso in sola lettura a tutte le risorse di Amazon Neptune Analytics insieme alle autorizzazioni di sola lettura per i servizi dipendenti.

Questa policy include le autorizzazioni per eseguire le seguenti operazioni:

- Per Amazon EC2: recuperare informazioni su VPC, sottoreti, gruppi di sicurezza e zone di disponibilità.
- Per AWS KMS: recupera informazioni su chiavi e alias KMS.
- Per CloudWatch: recupera informazioni sulle metriche. CloudWatch
- Per CloudWatch i log: recupera informazioni sui flussi di CloudWatch log e sugli eventi.

> **Note**
>
> Questa policy è stata rilasciata il 29 novembre 2023.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowReadOnlyPermissionsForNeptuneGraph",
      "Effect": "Allow",
      "Action": [
        "neptune-graph:Get*",
        "neptune-graph:List*",
        "neptune-graph:Read*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForEC2",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:DescribeAvailabilityZones"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForKMS",
      "Effect": "Allow",
      "Action": [
        "kms:ListKeys",
        "kms:ListAliases"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForCloudwatch",
```

```
      "Effect": "Allow",
      "Action": [
        "cloudwatch:GetMetricData",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForLogs",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams",
        "logs:GetLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/neptune/*:log-stream:*"
      ]
    }
  ]
}
```

Policy AWS gestita da **AWSServiceRoleForNeptuneGraphPolicy**

La politica AWSServiceRoleForNeptuneGraphPolicygestita riportata di seguito consente di accedere ai grafici per CloudWatch pubblicare metriche e registri operativi e di utilizzo. Per informazioni, consulta nan-service-linked-roles.

> **Note**
>
> Questa policy è stata rilasciata il 29 novembre 2023.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GraphMetrics",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
```

```
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": [
            "AWS/Neptune",
            "AWS/Usage"
          ]
        }
      }
    },
    {
      "Sid": "GraphLogGroup",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/neptune/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
      }
    },
    {
      "Sid": "GraphLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/neptune/*:log-stream:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
      }
    }
  ]
```

```
}
```

Chiavi di contesto delle condizioni IAM supportate da Amazon Neptune

Puoi specificare le condizioni nelle policy IAM che controllano l'accesso alle azioni e alle risorse di gestione di Neptune. La dichiarazione di policy diventa effettiva solo quando le condizioni sono true.

Ad esempio, potresti volere che una dichiarazione di policy diventi effettiva solo dopo una data specifica o che consenta l'accesso solo quando nella richiesta API è presente un valore specifico.

Per specificare le condizioni, è possibile utilizzare le chiavi di condizione predefinite nell'elemento `Condition` di una dichiarazione di policy insieme agli operatori della policy della condizione IAM come uguale a o minore di.

Se specifichi più elementi `Condition` in un'istruzione o più chiavi in un singolo elemento `Condition`, questi vengono valutati da AWS utilizzando un'operazione `AND` logica. Se si specificano più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'operazione logica. `OR` Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

Puoi anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, puoi autorizzare un utente IAM ad accedere a una risorsa solo se è stata taggata con il relativo nome utente IAM. Per ulteriori informazioni, consulta Elementi delle policy IAM: variabili e tag nella Guida per l'utente di IAM.

Il tipo di dati di una chiave di condizione determina quali operatori di condizione è possibile utilizzare per confrontare i valori nella richiesta con i valori della dichiarazione di policy. Se si utilizza un operatore di condizione che non è compatibile con tale tipo di dati, la corrispondenza ha sempre esito negativo e la dichiarazione di policy non viene mai applicata.

Chiavi di condizione IAM per le dichiarazioni di policy amministrative di Neptune

- Chiavi di condizione globali: è possibile utilizzare la maggior parte delle chiavi di condizione AWS globali nelle dichiarazioni delle politiche amministrative di Neptune.
- Chiavi di condizione specifiche del servizio: si tratta di chiavi definite per servizi specifici. AWS Quelle supportate da Neptune per le dichiarazioni di policy amministrative sono elencate in Chiavi di condizione disponibili nelle dichiarazioni di policy amministrative IAM di Neptune.

Chiavi di condizione IAM per le dichiarazioni di policy di accesso ai dati

- Chiavi di condizione globali: il sottoinsieme di queste chiavi supportate da Neptune nelle dichiarazioni di policy di accesso ai dati è elencato in AWS chiavi di contesto delle condizioni globali supportate da Neptune nelle dichiarazioni sulle politiche di accesso ai dati.
- Le chiavi di condizione specifiche del servizio che Neptune definisce per le dichiarazioni di policy di accesso ai dati sono elencate in Chiavi di condizione.

Dichiarazioni di policy amministrative IAM personalizzate per Amazon Neptune

Le dichiarazioni di policy amministrative consentono di controllare cosa può fare un utente IAM per gestire un database Neptune.

Una dichiarazione di policy amministrativa di Neptune garantisce l'accesso a una o più azioni amministrative e risorse amministrative supportate da Neptune. È inoltre possibile utilizzare le Chiavi di condizione per rendere più specifiche le autorizzazioni amministrative.

> **Note**
>
> Poiché Neptune condivide funzionalità con Amazon RDS, le azioni amministrative, le risorse e le chiavi di condizione specifiche del servizio nelle dichiarazioni di policy amministrative utilizzano un prefisso `rds:` per impostazione predefinita.

Argomenti

- Azioni disponibili nelle dichiarazioni di policy amministrative IAM di Neptune
- Tipi di risorse disponibili nelle dichiarazioni di policy amministrative IAM di Neptune
- Chiavi di condizione disponibili nelle dichiarazioni di policy amministrative IAM di Neptune
- Esempi di dichiarazioni di policy amministrative IAM per Neptune

Azioni disponibili nelle dichiarazioni di policy amministrative IAM di Neptune

È possibile utilizzare le azioni amministrative elencate di seguito nell'elemento `Action` di una dichiarazione di policy IAM per controllare l'accesso alle API di gestione Neptune. Quando utilizzi un'operazione in una policy, in genere consenti o rifiuti l'accesso all'operazione API o al comando CLI

con lo stesso nome. Tuttavia, in alcuni casi, una singola operazione controlla l'accesso a più di una operazione. In alternativa, alcune operazioni richiedono operazioni differenti.

Il campo `Resource type` nell'elenco seguente indica se ogni azione supporta le autorizzazioni a livello di risorsa. Se questo campo non contiene alcun valore, è necessario specificare tutte le risorse ("*") nell'elemento `Resource` della dichiarazione di policy. Se la colonna include un tipo di risorsa, è possibile specificare un ARN della risorsa di quel tipo in una dichiarazione con tale azione. I tipi di risorse amministrative di Neptune sono elencati in questa pagina.

Le risorse necessarie sono indicate nell'elenco sottostante con un asterisco (*). Se specifichi un ARN di autorizzazione a livello di risorsa in una dichiarazione utilizzando questa operazione, allora deve essere di questo tipo. Alcune operazioni supportano più tipi di risorse. Se un tipo di risorsa è facoltativo (in altre parole, non è contrassegnato da un asterisco), non è necessario includerlo.

Per ulteriori informazioni sui campi elencati qui, consulta la tabella delle azioni nella Guida per l'utente di IAM.

rds: DBCluster AddRoleTo

`AddRoleToDBCluster` associa un ruolo IAM a un cluster database Neptune.

Livello di accesso: `Write`.

Azioni dipendenti: `iam:PassRole`.

Tipo di risorsa: cluster (obbligatorio).

rds: AddSourceIdentifierToSubscription

`AddSourceIdentifierToSubscription` aggiunge un identificatore di origine a una sottoscrizione alle notifiche di eventi Neptune esistente.

Livello di accesso: `Write`.

Tipo di risorsa: es (obbligatorio)

rds: AddTagsToResource

`AddTagsToResource` associa un ruolo IAM a un cluster database Neptune.

Livello di accesso: `Write`.

Tipi di risorsa:

- db
- es
- pg
- cluster-snapshot
- subgrp

Chiavi di condizione:

- *aws:RequestTag//tag-key*
- aws: TagKeys

rds: ApplyPendingMaintenanceAction

`ApplyPendingMaintenanceAction` applica un'azione di manutenzione in sospeso a una risorsa.

Livello di accesso: `Write`.

Tipo di risorsa: db (obbligatorio).

RDS: copydb ClusterParameterGroup

`CopyDBClusterParameterGroup` copia il gruppo di parametri del cluster database specificato.

Livello di accesso: `Write`.

Tipo di risorsa: cluster-pg (obbligatorio).

RDS: copyDB ClusterSnapshot

`CopyDBClusterSnapshot` copia uno snapshot di un cluster database.

Livello di accesso: `Write`.

Tipo di risorsa: cluster-snapshot (obbligatorio).

RDS: copyDB ParameterGroup

`CopyDBParameterGroup` copia il gruppo di parametri database specificato.

Livello di accesso: `Write`.

Tipo di risorsa: pg (obbligatorio).

rds:CreateDBCluster

`CreateDBCluster` crea un nuovo cluster database Neptune.

Livello di accesso: `Tagging`.

Azioni dipendenti: `iam:PassRole`.

Tipi di risorsa:

- cluster (obbligatorio).
- cluster-pg (obbligatorio).
- subgrp (obbligatorio).

Chiavi di condizione:

- *aws://tag-key RequestTag*
- aws: TagKeys
- nettuno-rds_ DatabaseEngine

RDS: creato B ClusterParameterGroup

`CreateDBClusterParameterGroup` crea un nuovo gruppo di parametri del cluster database.

Livello di accesso: `Tagging`.

Tipo di risorsa: cluster-pg (obbligatorio).

Chiavi di condizione:

- *aws://tag-key RequestTag*
- aws: TagKeys

RDS: creato B ClusterSnapshot

`CreateDBClusterSnapshot` crea uno snapshot di un cluster database.

Livello di accesso: `Tagging`.

Tipi di risorsa:

- cluster (obbligatorio).
- cluster-snapshot (obbligatorio).

Chiavi di condizione:

- *aws://tag-key RequestTag*
- aws: TagKeys

rds:CreateDBInstance

`CreateDBInstance`: crea una nuova istanza database.

Livello di accesso: `Tagging`.

Azioni dipendenti: `iam:PassRole`.

Tipi di risorsa:

- db (obbligatorio).
- pg (obbligatorio).
- subgrp (obbligatorio).

Chiavi di condizione:

- aws:RequestTag//*tag-key*
- aws: TagKeys

RDS: creato B ParameterGroup

`CreateDBParameterGroup` crea un nuovo gruppo di parametri database.

Livello di accesso: `Tagging`.

Tipo di risorsa: pg (obbligatorio).

Chiavi di condizione:

- *aws://tag-key RequestTag*
- aws: TagKeys

RDS: creato B SubnetGroup

CreateDBSubnetGroup crea un nuovo gruppo di sottoreti del database.

Livello di accesso: Tagging.

Tipo di risorsa: subgrp (obbligatorio).

Chiavi di condizione:

- *aws://tag-key RequestTag*
- aws: TagKeys

rds: CreateEventSubscription

CreateEventSubscription crea una sottoscrizione alle notifiche di eventi Neptune.

Livello di accesso: Tagging.

Tipo di risorsa: es (obbligatorio)

Chiavi di condizione:

- *aws:RequestTag//tag-key*
- aws: TagKeys

rds:DeleteDBCluster

DeleteDBCluster elimina un cluster database Neptune esistente.

Livello di accesso: Write.

Tipi di risorsa:

- cluster (obbligatorio).

- cluster-snapshot (obbligatorio).

RDS: elimina DB ClusterParameterGroup

DeleteDBClusterParameterGroup elimina un gruppo di parametri del cluster database specificato.

Livello di accesso: `Write`.

Tipo di risorsa: cluster-pg (obbligatorio).

RDS: elimina DB ClusterSnapshot

DeleteDBClusterSnapshot elimina uno snapshot del cluster database.

Livello di accesso: `Write`.

Tipo di risorsa: cluster-snapshot (obbligatorio).

rds:DeleteDBInstance

DeleteDBInstance elimina un'istanza database specificata.

Livello di accesso: `Write`.

Tipo di risorsa: db (obbligatorio).

RDS: elimina DB ParameterGroup

DeleteDBParameterGroupelimina un DB specificato. ParameterGroup

Livello di accesso: `Write`.

Tipo di risorsa: pg (obbligatorio).

RDS: delete DB SubnetGroup

DeleteDBSubnetGroup elimina un gruppo di sottoreti del database.

Livello di accesso: `Write`.

Tipo di risorsa: subgrp (obbligatorio).

rds: DeleteEventSubscription

`DeleteEventSubscription` elimina una sottoscrizione alle notifiche di eventi.

Livello di accesso: `Write`.

Tipo di risorsa: es (obbligatorio)

RDS: descritto B ClusterParameterGroups

`DescribeDBClusterParameterGroups`restituisce un elenco di descrizioni DB. ClusterParameterGroup

Livello di accesso: `List`.

Tipo di risorsa: cluster-pg (obbligatorio).

RDS: DescribedB ClusterParameters

`DescribeDBClusterParameters` restituisce l'elenco dettagliato di parametri per un determinato gruppo di parametri del cluster database.

Livello di accesso: `List`.

Tipo di risorsa: cluster-pg (obbligatorio).

RDS: descritto B ClusterSnapshotAttributes

`DescribeDBClusterSnapshotAttributes` restituisce un elenco di nomi e valori degli attributi dello snapshot del cluster database per uno snapshot del cluster database manuale.

Livello di accesso: `List`.

Tipo di risorsa: cluster-snapshot (obbligatorio).

RDS: descritto B ClusterSnapshots

`DescribeDBClusterSnapshots` restituisce informazioni sugli snapshot del cluster database.

Livello di accesso: `Read`.

rds:DescribeDBClusters

`DescribeDBClusters` restituisce informazioni su un cluster database Neptune di cui è stato effettuato il provisioning.

Livello di accesso: `List`.

Tipo di risorsa: cluster (obbligatorio).

RDS: descritto B EngineVersions

`DescribeDBEngineVersions` restituisce un elenco dei motori di database disponibili.

Livello di accesso: `List`.

Tipo di risorsa: pg (obbligatorio).

rds:DescribeDBInstances

`DescribeDBInstances` restituisce informazioni sulle istanze database.

Livello di accesso: `List`.

Tipo di risorsa: es (obbligatorio)

RDS: descritto B ParameterGroups

`DescribeDBParameterGroups`restituisce un elenco di descrizioni DB. ParameterGroup

Livello di accesso: `List`.

Tipo di risorsa: pg (obbligatorio).

rds:DescribeDBParameters

`DescribeDBParameters` restituisce un elenco dettagliato di parametri per un determinato gruppo di parametri database.

Livello di accesso: `List`.

Tipo di risorsa: pg (obbligatorio).

RDS: DescribedB SubnetGroups

`DescribeDBSubnetGroups`restituisce un elenco di descrizioni DB. SubnetGroup

Livello di accesso: `List`.

Tipo di risorsa: subgrp (obbligatorio).

rds: DescribeEventCategories

`DescribeEventCategories` restituisce un elenco di categorie per tutti i tipi di origine eventi oppure, se specificato, per un determinato tipo di origine.

Livello di accesso: `List`.

rds: DescribeEventSubscriptions

`DescribeEventSubscriptions` elenca tutte le descrizioni di sottoscrizioni per un account cliente.

Livello di accesso: `List`.

Tipo di risorsa: `es` (obbligatorio)

rds: DescribeEvents

`DescribeEvents` restituisce eventi relativi a istanze database, gruppi di sicurezza del database e gruppi di parametri database degli ultimi 14 giorni.

Livello di accesso: `List`.

Tipo di risorsa: `es` (obbligatorio)

rds: DB DescribeOrderable InstanceOptions

`DescribeOrderableDBInstanceOptions` restituisce un elenco delle opzioni delle istanze database ordinabili per il motore specificato.

Livello di accesso: `List`.

rds: DescribePendingMaintenanceActions

`DescribePendingMaintenanceActions` restituisce un elenco di risorse (ad esempio, istanze database) che hanno almeno un'operazione di manutenzione in sospeso.

Livello di accesso: `List`.

Tipo di risorsa: `db` (obbligatorio).

rds: DB DescribeValid InstanceModifications

`DescribeValidDBInstanceModifications` elenca le modifiche disponibili che è possibile apportare all'istanza database.

Livello di accesso: `List`.

Tipo di risorsa: db (obbligatorio).

rds:FailoverDBCluster

`FailoverDBCluster` forza un failover per un cluster database.

Livello di accesso: `Write`.

Tipo di risorsa: cluster (obbligatorio).

rds: ListTagsForResource

`ListTagsForResource` elenca tutti i tag su una risorsa Neptune.

Livello di accesso: `Read`.

Tipi di risorsa:

- cluster-snapshot
- db
- es
- pg
- subgrp

rds:ModifyDBCluster

`ModifyDBCluster`

Modifica un'impostazione per un cluster database Neptune.

Livello di accesso: `Write`.

Azioni dipendenti: `iam:PassRole`.

Tipi di risorsa:

- cluster (obbligatorio).
- cluster-pg (obbligatorio).

RDS: modifica DB ClusterParameterGroup

`ModifyDBClusterParameterGroup` modifica i parametri di un gruppo di parametri del cluster database.

Livello di accesso: `Write`.

Tipo di risorsa: cluster-pg (obbligatorio).

RDS: ModifyDB ClusterSnapshotAttribute

`ModifyDBClusterSnapshotAttribute` aggiunge un attributo e i valori a uno snapshot del cluster database manuale o li rimuove dallo stesso.

Livello di accesso: `Write`.

Tipo di risorsa: cluster-snapshot (obbligatorio).

rds:ModifyDBInstance

`ModifyDBInstance` modifica le impostazioni di un'istanza database.

Livello di accesso: `Write`.

Azioni dipendenti: `iam:PassRole`.

Tipi di risorsa:

- db (obbligatorio).
- pg (obbligatorio).

RDS: ModifyDB ParameterGroup

`ModifyDBParameterGroup` modifica i parametri di un gruppo di parametri database.

Livello di accesso: `Write`.

Tipo di risorsa: pg (obbligatorio).

RDS: ModifyDB SubnetGroup

`ModifyDBSubnetGroup` modifica un gruppo di sottoreti del database esistente.

Livello di accesso: `Write`.

Tipo di risorsa: subgrp (obbligatorio).

rds: ModifyEventSubscription

`ModifyEventSubscription` modifica una sottoscrizione alle notifiche di eventi Neptune esistente.

Livello di accesso: `Write`.

Tipo di risorsa: es (obbligatorio)

rds:RebootDBInstance

`RebootDBInstance` riavvia il servizio del motore di database per l'istanza.

Livello di accesso: `Write`.

Tipo di risorsa: db (obbligatorio).

rds: DBCluster RemoveRoleFrom

`RemoveRoleFromDBCluster`dissocia un ruolo AWS Identity and Access Management (IAM) da un cluster Amazon Neptune DB.

Livello di accesso: `Write`.

Azioni dipendenti: `iam:PassRole`.

Tipo di risorsa: cluster (obbligatorio).

aggiunge: RemoveSourceIdentifierFromSubscription

`RemoveSourceIdentifierFromSubscription` rimuove un identificatore di origine da una sottoscrizione alle notifiche di eventi Neptune esistente.

Livello di accesso: `Write`.

Tipo di risorsa: es (obbligatorio)

rds: RemoveTagsFromResource

`RemoveTagsFromResource` rimuove tag di metadati da una risorsa Neptune.

Livello di accesso: `Tagging`.

Tipi di risorsa:

- cluster-snapshot
- db
- es
- pg
- subgrp

Chiavi di condizione:

- *aws:RequestTag//tag-key*
- aws: TagKeys

RDS: resetDB ClusterParameterGroup

`ResetDBClusterParameterGroup` modifica i parametri di un gruppo di parametri del cluster database con il valore predefinito.

Livello di accesso: `Write`.

Tipo di risorsa: cluster-pg (obbligatorio).

RDS: resetDB ParameterGroup

`ResetDBParameterGroup` modifica i parametri di un gruppo di parametri database con il valore predefinito del motore/sistema.

Livello di accesso: `Write`.

Tipo di risorsa: pg (obbligatorio).

RDS: ripristina DB ClusterFromSnapshot

`RestoreDBClusterFromSnapshot` crea un nuovo cluster database da uno snapshot del cluster database.

Livello di accesso: `Write`.

Azioni dipendenti: `iam:PassRole`.

Tipi di risorsa:

- cluster (obbligatorio).
- cluster-snapshot (obbligatorio).

Chiavi di condizione:

- *aws://tag-key RequestTag*
- aws: TagKeys

RDS: ripristina DB ClusterToPointInTime

`RestoreDBClusterToPointInTime` ripristina un cluster database a un point-in-time arbitrario.

Livello di accesso: `Write`.

Azioni dipendenti: `iam:PassRole`.

Tipi di risorsa:

- cluster (obbligatorio).
- subgrp (obbligatorio).

Chiavi di condizione:

- *aws://tag-key RequestTag*
- aws: TagKeys

rds:StartDBCluster

`StartDbCluster` avvia il cluster database specificato.

Livello di accesso: `Write`.

Tipo di risorsa: cluster (obbligatorio).

rds:StopDBCluster

`StopDBCluster` arresta il cluster database specificato.

Livello di accesso: `Write`.

Tipo di risorsa: cluster (obbligatorio).

Tipi di risorse disponibili nelle dichiarazioni di policy amministrative IAM di Neptune

Neptune supporta i tipi di risorse riportati nella tabella seguente da utilizzare nell'elemento `Resource` delle dichiarazioni di policy di amministrazione IAM. Per ulteriori informazioni sull'elemento `Resource`, consulta Elementi JSON della policy IAM: Risorsa.

L'elenco delle azioni di amministrazione di Neptune identifica i tipi di risorse che è possibile specificare con ogni azione. Un tipo di risorsa determina anche quali chiavi di condizione è possibile includere in una policy, come specificato nell'ultima colonna della tabella seguente.

La colonna ARN della tabella seguente specifica il formato del nome della risorsa Amazon (ARN) che è necessario utilizzare per fare riferimento alle risorse di questo tipo. Le porzioni precedute da un `$` devono essere sostituite con i valori effettivi per il proprio scenario. Ad esempio, se è presente `$user-name` in un ARN, è necessario sostituire tale stringa con l'effettivo nome dell'utente IAM o con una variabile della policy che contiene il nome di un utente IAM. Per ulteriori informazioni sugli ARN, consulta ARN IAM e Utilizzo degli ARN amministrativi in Amazon Neptune.

La colonna `Condition Keys` specifica le chiavi di contesto delle condizioni che è possibile includere in una dichiarazione di policy IAM solo quando sia questa risorsa che un'azione di supporto compatibile sono incluse nella dichiarazione.

Tipi di risorsa	ARN	Chiavi di condizione
`cluster` (cluster database)	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :cluster: `*`instance-name`*	aws:ResourceTag//*`tag-key`* rds:cluster-tag/*`tag-key`*
`cluster-pg` (gruppo di parametri del cluster database)	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :cluster-pg: `*`neptune-DBClusterParameterGroupName`*	*`aws://chiave-tag ResourceTag`*

Tipi di risorsa	ARN	Chiavi di condizione
cluster-snapshot (snapshot del cluster database)	arn:*partition* :rds:*region*:*account-id* :cluster-snapshot: *neptune-DBClusterSnapshotName*	*aws://chiave-tag ResourceTag* *rds:/chiave-tag cluster-snapshot-tag*
db (istanza database)	arn:*partition* :rds:*region*:*account-id* :db:*neptune-DbInstanceName*	*aws:/chiave-tag ResourceTag* rds: DatabaseClass rds: DatabaseEngine rds:db-tag/*tag-key*
es (sottoscrizione a eventi)	arn:*partition* :rds:*region*:*account-id* :es:*neptune-CustSubscriptionId*	*aws:ResourceTag//tag-key* rds:es-tag/*tag-key*
pg (gruppo di parametri database)	arn:*partition* :rds:*region*:*account-id* :pg:*neptune-ParameterGroupName*	*aws://chiave-tag ResourceTag* rds:pg-tag/*tag-key*
subgrp (gruppo di sottoreti del database)	arn:*partition* :rds:*region*:*account-id* :subgrp:*neptune-DBSubnetGroupName* }	*aws://chiave-tag ResourceTag* rds:subgrp-tag/*tag-key*

Chiavi di condizione disponibili nelle dichiarazioni di policy amministrative IAM di Neptune

Utilizzando chiavi di condizione, è possibile specificare le condizioni in una dichiarazione di policy IAM in modo che la dichiarazione diventi effettiva solo quando le condizioni sono vere. Le chiavi di condizione che è possibile utilizzare nelle dichiarazioni di policy amministrative di Neptune rientrano nelle seguenti categorie:

- Chiavi di condizione globali: sono definite da AWS per l'uso generale con AWS i servizi. La maggior parte può essere utilizzata nelle dichiarazioni di policy amministrative di Neptune.
- Chiavi di condizione delle proprietà delle risorse amministrative: queste chiavi, elencate di seguito, si basano sulle proprietà delle risorse amministrative.
- Chiavi di condizione di accesso basate su tag: queste chiavi, elencate di seguito, si basano sui tagAWS collegati alle risorse amministrative.

Chiavi di condizione delle proprietà delle risorse amministrative di Neptune

Chiavi di condizione	Descrizione	Type
`rds:DatabaseClass`	Filtra l'accesso in base al tipo di classe di istanza database.	Stringa
`rds:DatabaseEngine`	Filtra l'accesso in base al motore di database. Per i valori possibili, fai riferimento al parametro del motore nell'API CreateDBInstance	Stringa
`rds:DatabaseName`	Filtra l'accesso in base al nome definito dall'utente del database nell'istanza database	Stringa
`rds:EndpointType`	Filtra l'accesso in base al tipo di endpoint. Uno di: di LETTURA, di SCRITTURA e PERSONALIZZATO.	Stringa
`rds:Vpc`	Filtra l'accesso in base al valore che specifica se l'istanza database viene eseguita in un Amazon Virtual Private Cloud (Amazon VPC). Per indicare che l'istanza database viene eseguita in Amazon VPC, specifica `true`.	Booleano

Chiavi di condizione amministrative basate su tag

Amazon Neptune permette di specificare le condizioni in una policy IAM utilizzando tag personalizzati per controllare l'accesso a Neptune tramite Documentazione di riferimento delle API di gestione.

Ad esempio, se aggiungi un tag denominato `environment` alle tue istanze database, con valori come `beta`, `staging` e `production`, puoi quindi creare una policy che limiti l'accesso alle istanze in base al valore di quel tag.

Important

Se gestisci l'accesso alle risorse Neptune utilizzando i tag, assicurati di proteggere l'accesso ai tag. Puoi limitare l'accesso ai tag creando policy per le azioni `AddTagsToResource` e `RemoveTagsFromResource`.
Ad esempio, potresti usare la policy seguente per negare agli utenti la possibilità di aggiungere o rimuovere tag per tutte le risorse. Potresti quindi creare policy per permettere a utenti specifici di aggiungere o rimuovere tag.

```
{ "Version": "2012-10-17",
  "Statement":[
    { "Sid": "DenyTagUpdates",
      "Effect": "Deny",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource":"*"
    }
  ]
}
```

Le seguenti chiavi di condizione basate su tag funzionano solo con le risorse amministrative contenute nelle dichiarazioni di policy amministrative.

Chiavi di condizione amministrative basate su tag

Chiavi di condizione	Descrizione	Type
aws:RequestTag/${TagKey}	Filtra l'accesso in base alla presenza di coppie chiave-valore di tag nella richiesta.	Stringa
aws:ResourceTag/${TagKey}	Filtra l'accesso in base a coppie chiave/valore di tag collegate alla risorsa.	Stringa
aws:TagKeys	Filtra l'accesso in base alla presenza di chiavi di tag nella richiesta.	Stringa
rds:cluster-pg-tag/${TagKey}	Filtra l'accesso in base al tag collegato a un gruppo di parametri del cluster database.	Stringa
rds:cluster-snapshot-tag/${TagKey}	Filtra l'accesso in base al tag collegato a uno snapshot del cluster database.	Stringa
rds:cluster-tag/${TagKey}	Filtra l'accesso in base al tag collegato a un cluster database.	Stringa
rds:db-tag/${TagKey}	Filtra l'accesso in base al tag collegato a un'istanza database.	Stringa
rds:es-tag/${TagKey}	Filtra l'accesso in base al tag collegato a una sottoscrizione di eventi.	Stringa
rds:pg-tag/${TagKey}	Filtra l'accesso in base al tag collegato a un gruppo di parametri database.	Stringa

Chiavi di condizione	Descrizione	Type
`rds:req-tag/${TagKey}`	Filtra l'accesso in base al set di chiavi e valori di tag che possono essere utilizzati per aggiungere tag a una risorsa.	Stringa
`rds:secgrp-tag/${TagKey}`	Filtra l'accesso in base al tag collegato a un gruppo di sicurezza database.	Stringa
`rds:snapshot-tag/${TagKey}`	Filtra l'accesso in base al tag collegato a uno snapshot del database.	Stringa
`rds:subgrp-tag/${TagKey}`	Filtra l'accesso in base al tag collegato a un gruppo di sottoreti DB	Stringa

Esempi di dichiarazioni di policy amministrative IAM per Neptune

Esempi di policy amministrative generali

Gli esempi seguenti mostrano come creare policy amministrative di Neptune che concedono le autorizzazioni per eseguire varie azioni di gestione su un cluster database.

Policy che impedisce a un utente IAM di eliminare un'istanza database specificata

Di seguito è riportato un esempio di policy che impedisce a un utente IAM di eliminare un'istanza database Neptune specificata:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyDeleteOneInstance",
      "Effect": "Deny",
      "Action": "rds:DeleteDBInstance",
      "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-instance-name"
    }
```

```
  ]
}
```

Policy che concede l'autorizzazione per creare nuove istanze database

Di seguito è riportato un esempio di policy che consente a un utente IAM di creare istanze database in un cluster database Neptune specificato:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateInstance",
      "Effect": "Allow",
      "Action": "rds:CreateDBInstance",
      "Resource": "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster"
    }
  ]
}
```

Policy che concede l'autorizzazione per creare nuove istanze database che utilizzano un gruppo di parametri database specifico

Di seguito è riportato un esempio di policy che consente a un utente IAM di creare istanze database in un cluster database specificato (`us-west-2` in questo esempio) utilizzando solo un gruppo di parametri database specificato.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateInstanceWithPG",
      "Effect": "Allow",
      "Action": "rds:CreateDBInstance",
      "Resource": [
        "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster",
        "arn:aws:rds:us-west-2:123456789012:pg:my-instance-pg"
      ]
    }
  ]
}
```

Policy che concede l'autorizzazione per descrivere una risorsa

Di seguito è riportato un esempio di policy che consente a un utente IAM di descrivere qualsiasi risorsa Neptune.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDescribe",
      "Effect": "Allow",
      "Action": "rds:Describe*",
      "Resource": *
    }
  ]
}
```

Esempi di policy amministrative basate su tag

Gli esempi seguenti mostrano come creare policy amministrative di Neptune che utilizzano tag per filtrare le autorizzazioni per varie azioni di gestione su un cluster database.

Esempio 1: Concedere l'autorizzazione per le azioni su una risorsa tramite un tag personalizzato che può assumere più valori

La policy seguente consente l'uso dell'API`ModifyDBInstance`, `CreateDBInstance` o `DeleteDBInstance` su qualsiasi istanza database con il tag `env` impostato su un `dev` o `test`:

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDevTestAccess",
      "Effect": "Allow",
      "Action": [
        "rds:ModifyDBInstance",
        "rds:CreateDBInstance",
        "rds:DeleteDBInstance"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "rds:db-tag/env": [
```

```
                "dev",
                "test"
            ],
            "rds:DatabaseEngine": "neptune"
          }
        }
      }
    ]
}
```

Esempio 2: Limitare il set di chiavi e valori di tag che possono essere utilizzati per aggiungere tag a una risorsa

Questa policy utilizza una chiave `Condition` per consentire a un tag con la chiave `env` e il valore `test`, `qa` o `dev` di essere aggiunto a una risorsa:

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowTagAccessForDevResources",
      "Effect": "Allow",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "rds:req-tag/env": [
            "test",
            "qa",
            "dev"
          ],
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Esempio 3: Consentire l'accesso completo alle risorse Neptune in base a **`aws:ResourceTag`**

La policy seguente è simile al primo esempio, ma utilizza `aws:ResourceTag`:

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFullAccessToDev",
      "Effect": "Allow",
      "Action": [
        "rds:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/env": "dev",
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Dichiarazioni di policy di accesso ai dati IAM personalizzate per Amazon Neptune

Le dichiarazioni di policy di accesso ai dati di Neptune utilizzano azioni di accesso ai dati, risorse e chiavi di condizione, tutte precedute dal prefisso `neptune-db:`.

Argomenti

- Utilizzo delle azioni di query nelle dichiarazioni di policy di accesso ai dati di Neptune
- Azioni disponibili nelle dichiarazioni di policy di accesso ai dati IAM di Neptune
- Specifica delle risorse nelle dichiarazioni di policy di accesso ai dati IAM di Neptune
- Chiavi di condizione disponibili nelle dichiarazioni di policy di accesso ai dati IAM di Neptune
- Esempi di policy di accesso ai dati IAM di Neptune

Utilizzo delle azioni di query nelle dichiarazioni di policy di accesso ai dati di Neptune

Esistono tre azioni di query di Neptune che possono essere utilizzate nelle dichiarazioni di policy di accesso ai dati, vale a dire `ReadDataViaQuery`, `WriteDataViaQuery` e `DeleteDataViaQuery`. Una particolare query può richiedere le autorizzazioni per eseguire più di una di queste azioni e

potrebbe non essere sempre ovvio quale combinazione di queste azioni debba essere consentita per eseguire una query.

Prima di eseguire una query, Neptune determina le autorizzazioni necessarie per eseguire ogni passaggio della query e le combina nel set completo di autorizzazioni richieste dalla query. Nota che questo set completo di autorizzazioni include tutte le azioni che la query potrebbe eseguire, il che non è necessariamente il set di azioni che la query effettivamente eseguirà quando verrà eseguita sui dati.

Ciò significa che per consentire l'esecuzione di una determinata query, è necessario fornire le autorizzazioni per ogni azione che la query potrebbe eseguire, indipendentemente dal fatto che le esegua effettivamente o meno.

Ecco alcuni esempi di query Gremlin in cui ciò viene spiegato più dettagliatamente:

- ```
  g.V().count()
  ```

  `g.V()` e `count()` richiedono solo l'accesso in lettura, quindi la query nel suo complesso richiede solo l'accesso `ReadDataViaQuery`.

- ```
  g.addV()
  ```

  `addV()` deve verificare se esiste o meno un vertice con un determinato ID prima di inserirne uno nuovo. Ciò significa che richiede sia l'accesso `ReadDataViaQuery` che `WriteDataViaQuery`.

- ```
  g.V('1').as('a').out('created').addE('createdBy').to('a')
  ```

  `g.V('1').as('a')` e `out('created')` richiedono solo l'accesso in lettura, ma `addE().from('a')` richiede sia l'accesso in lettura che in scrittura perché `addE()` deve leggere i vertici `from` e `to` e verificare se esiste già un arco con lo stesso ID prima di aggiungerne uno nuovo. La query nel suo complesso necessita quindi sia dell'accesso `ReadDataViaQuery` che `WriteDataViaQuery`.

- ```
  g.V().drop()
  ```

  `g.V()` richiede solo l'accesso in lettura. `drop()` necessita sia dell'accesso in lettura che di quello di eliminazione perché deve leggere un vertice o uno arco prima di eliminarlo, quindi la query nel suo complesso richiede sia l'accesso `ReadDataViaQuery` che `DeleteDataViaQuery`.

- ```
  g.V('1').property(single, 'key1', 'value1')
  ```

  `g.V('1')` richiede solo l'accesso in lettura, ma `property(single, 'key1', 'value1')` richiede l'accesso in lettura, scrittura ed eliminazione. In questo caso, il passaggio `property()` inserisce la chiave e il valore se non esistono già nel vertice, ma se esistono già, elimina il valore della proprietà esistente e inserisce un nuovo valore al suo posto. Pertanto, la query nel suo complesso richiede l'accesso `ReadDataViaQuery`, `WriteDataViaQuery` e `DeleteDataViaQuery`.

  Qualsiasi query che contenga un passaggio `property()` avrà bisogno delle autorizzazioni `ReadDataViaQuery`, `WriteDataViaQuery` e `DeleteDataViaQuery`.

Ecco alcuni esempi di openCypher:

- ```
  MATCH (n)
  RETURN n
  ```

  Questa query legge tutti i nodi del database e li restituisce, il che richiede solo l'accesso `ReadDataViaQuery`.

- ```
  MATCH (n:Person)
  SET n.dept = 'AWS'
  ```

  Questa query richiede l'accesso `ReadDataViaQuery`, `WriteDataViaQuery` e `DeleteDataViaQuery`. Legge tutti i nodi con l'etichetta 'Person' e vi aggiunge una nuova proprietà con la chiave `dept` e il valore `AWS` oppure, se la proprietà `dept` esiste già, elimina il vecchio valore e inserisce `AWS` al suo posto. Inoltre, se il valore da impostare è `null`, `SET` elimina completamente la proprietà.

  Poiché in alcuni casi può essere necessario eliminare un valore esistente, la clausola `SET` necessita sempre delle autorizzazioni `DeleteDataViaQuery` nonché delle autorizzazioni `ReadDataViaQuery` e `WriteDataViaQuery`.

- ```
  MATCH (n:Person)
  DETACH DELETE n
  ```

Questa query richiede le autorizzazioni `ReadDataViaQuery` e `DeleteDataViaQuery`. Trova tutti i nodi con l'etichetta `Person` e li elimina insieme agli archi collegati a tali nodi e alle etichette e proprietà associate.

- ```
  MERGE (n:Person {name: 'John'})-[:knows]->(:Person {name: 'Peter'})
  RETURN n
  ```

  Questa query richiede le autorizzazioni `ReadDataViaQuery` e `WriteDataViaQuery`. La clausola `MERGE` cerca la corrispondenza con un modello specificato o lo crea. Poiché può verificarsi un'operazione di scrittura se non viene trovata una corrispondenza con il modello, sono necessarie le autorizzazioni di scrittura oltre a quelle di lettura.

Azioni disponibili nelle dichiarazioni di policy di accesso ai dati IAM di Neptune

Nota che le azioni di accesso ai dati di Neptune hanno il prefisso `neptune-db:`, mentre le azioni amministrative in Neptune hanno il prefisso `rds:`.

Il nome della risorsa Amazon (ARN) per una risorsa dati in IAM non è lo stesso ARN assegnato a un cluster in fase di creazione. È necessario creare l'ARN come mostrato in Specificare le risorse dati. Tali ARN di risorse dati possono utilizzare caratteri jolly per includere più risorse.

Le dichiarazioni sulla politica di accesso ai dati possono includere anche la chiave di QueryLanguage condizione neptune-db: per limitare l'accesso tramite il linguaggio di interrogazione.

A partire dal Rilascio: 1.2.0.0 (21/07/2022), Neptune supporta la limitazione delle autorizzazioni a una o più azioni specifiche di Neptune. Ciò fornisce un controllo degli accessi più granulare di quanto fosse possibile in precedenza.

> **Important**
>
> - Le modifiche apportate a una policy IAM richiedono fino a 10 minuti per essere applicate alle risorse Neptune specificate.
> - Le policy IAM applicate a un cluster database Neptune si applicano a tutte le istanze del cluster.
  ```

Azioni di accesso ai dati basate su query

> **Note**
>
> Non è sempre ovvio quali autorizzazioni siano necessarie per eseguire una determinata query, poiché le query possono potenzialmente eseguire più di un'azione a seconda dei dati che elaborano. Per ulteriori informazioni, consulta Utilizzo delle azioni di query.

`neptune-db:ReadDataViaQuery`

`ReadDataViaQuery` consente all'utente di leggere i dati dal database Neptune inviando query.

Gruppi di azioni: sola lettura, lettura-scrittura.

Chiavi di contesto dell'azione: `neptune-db:QueryLanguage`.

Risorse necessarie: database.

`neptune-db:WriteDataViaQuery`

`WriteDataViaQuery` consente all'utente di scrivere i dati nel database Neptune inviando query.

Gruppi di azione: lettura-scrittura.

Chiavi di contesto dell'azione: `neptune-db:QueryLanguage`.

Risorse necessarie: database.

`neptune-db:DeleteDataViaQuery`

`DeleteDataViaQuery` consente all'utente di eliminare i dati dal database Neptune inviando query.

Gruppi di azione: lettura-scrittura.

Chiavi di contesto dell'azione: `neptune-db:QueryLanguage`.

Risorse necessarie: database.

`neptune-db:GetQueryStatus`

`GetQueryStatus` consente all'utente di controllare lo stato di tutte le query attive.

Gruppi di azioni: sola lettura, lettura-scrittura.

Chiavi di contesto dell'azione: `neptune-db:QueryLanguage`.

Risorse necessarie: database.

neptune-db:GetStreamRecords

`GetStreamRecords` consente all'utente di recuperare i record di flusso da Neptune.

Gruppi di azione: lettura-scrittura.

Chiavi di contesto dell'azione: `neptune-db:QueryLanguage`.

Risorse necessarie: database.

neptune-db:CancelQuery

`CancelQuery` consente all'utente di annullare una query.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

Azioni generali di accesso ai dati

neptune-db:GetEngineStatus

`GetEngineStatus` consente all'utente di controllare lo stato del motore Neptune.

Gruppi di azioni: sola lettura, lettura-scrittura.

Risorse necessarie: database.

neptune-db:GetStatisticsStatus

`GetStatisticsStatus` consente all'utente di controllare lo stato delle statistiche raccolte per il database.

Gruppi di azioni: sola lettura, lettura-scrittura.

Risorse necessarie: database.

neptune-db:GetGraphSummary

`GetGraphSummary` L'API di riepilogo del grafo consente di recuperare un riepilogo di sola lettura del grafo.

Gruppi di azioni: sola lettura, lettura-scrittura.

Risorse necessarie: database.

neptune-db:ManageStatistics

`ManageStatistics` consente all'utente di gestire la raccolta di statistiche per il database.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

neptune-db:DeleteStatistics

`DeleteStatistics` consente all'utente di eliminare tutte le statistiche del database.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

neptune-db:ResetDatabase

`ResetDatabase` consente all'utente di ottenere il token necessario per una reimpostazione e di reimpostare il database Neptune.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

Azioni di accesso ai dati dello strumento di caricamento in blocco

neptune-db:StartLoaderJob

`StartLoaderJob` consente all'utente di avviare un processo dello strumento di caricamento in blocco.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

neptune-db:GetLoaderJobStatus

`GetLoaderJobStatus` consente all'utente di controllare lo stato di un processo dello strumento di caricamento in blocco

Gruppi di azioni: sola lettura, lettura-scrittura.

Risorse necessarie: database.

neptune-db:ListLoaderJobs

`ListLoaderJobs` consente all'utente di elencare tutti i processi dello strumento di caricamento in blocco.

Gruppi di azioni: solo elenco, sola lettura, lettura-scrittura.

Risorse necessarie: database.

neptune-db:CancelLoaderJob

`CancelLoaderJob` consente all'utente di annullare un processo del loader.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

Azioni di accesso ai dati di machine learning

neptune-db:StartMLDataProcessingJob

`StartMLDataProcessingJob` consente a un utente di avviare un processo di elaborazione dati Neptune ML.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

neptune-db:StartMLModelTrainingJob

`StartMLModelTrainingJob` consente a un utente di avviare un processo di addestramento dei modelli ML.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

neptune-db:StartMLModelTransformJob

`StartMLModelTransformJob` consente a un utente di avviare un processo di trasformazione dei modelli ML.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

neptune-db:CreateMLEndpoint

`CreateMLEndpoint` consente a un utente di creare un endpoint Neptune ML.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

neptune-db:GetMLDataProcessingJobStatus

`GetMLDataProcessingJobStatus` consente a un utente di controllare lo stato di un processo di elaborazione dati Neptune ML.

Gruppi di azioni: sola lettura, lettura-scrittura.

Risorse necessarie: database.

neptune-db:GetMLModelTrainingJobStatus

`GetMLModelTrainingJobStatus` consente a un utente di controllare lo stato di un processo di addestramento dei modelli Neptune ML.

Gruppi di azioni: sola lettura, lettura-scrittura.

Risorse necessarie: database.

neptune-db:GetMLModelTransformJobStatus

`GetMLModelTransformJobStatus` consente a un utente di controllare lo stato di un processo di trasformazione dei modelli Neptune ML.

Gruppi di azioni: sola lettura, lettura-scrittura.

Risorse necessarie: database.

neptune-db:GetMLEndpointStatus

`GetMLEndpointStatus` consente a un utente di controllare lo stato di un endpoint Neptune ML.

Gruppi di azioni: sola lettura, lettura-scrittura.

Risorse necessarie: database.

neptune-db:ListMLDataProcessingJobs

`ListMLDataProcessingJobs` consente a un utente di elencare tutti i processi di elaborazione dati Neptune ML.

Gruppi di azioni: solo elenco, sola lettura, lettura-scrittura.

Risorse necessarie: database.

neptune-db:ListMLModelTrainingJobs

`ListMLModelTrainingJobs` consente a un utente di elencare tutti i processi di addestramento dei modelli Neptune ML.

Gruppi di azioni: solo elenco, sola lettura, lettura-scrittura.

Risorse necessarie: database.

neptune-db:ListMLModelTransformJobs

`ListMLModelTransformJobs` consente a un utente di elencare tutti i processi di trasformazione dei modelli ML.

Gruppi di azioni: solo elenco, sola lettura, lettura-scrittura.

Risorse necessarie: database.

neptune-db:ListMLEndpoints

`ListMLEndpoints` consente a un utente di elencare tutti gli endpoint Neptune ML.

Gruppi di azioni: solo elenco, sola lettura, lettura-scrittura.

Risorse necessarie: database.

neptune-db:CancelMLDataProcessingJob

`CancelMLDataProcessingJob` consente a un utente di annullare un processo di elaborazione dati Neptune ML.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

`neptune-db:CancelMLModelTrainingJob`

`CancelMLModelTrainingJob` consente a un utente di annullare un processo di addestramento dei modelli Neptune ML.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

`neptune-db:CancelMLModelTransformJob`

`CancelMLModelTransformJob` consente a un utente di annullare un processo di trasformazione dei modelli Neptune ML.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

`neptune-db:DeleteMLEndpoint`

`DeleteMLEndpoint` consente a un utente di eliminare un endpoint Neptune ML.

Gruppi di azione: lettura-scrittura.

Risorse necessarie: database.

Specifica delle risorse nelle dichiarazioni di policy di accesso ai dati IAM di Neptune

Le risorse dati, come le azioni dati, hanno il prefisso `neptune-db:`.

In una policy di accesso ai dati di Neptune, è possibile specificare il cluster database a cui si concede l'accesso in un ARN con il seguente formato:

```
arn:aws:neptune-db:region:account-id:cluster-resource-id/*
```

Tale ARN della risorsa contiene le seguenti parti:

- *`region`*è la AWS regione per il cluster Amazon Neptune DB.
- *`account-id`* è il numero dell'account AWS per il cluster database.

- *cluster-resource-id* è l'ID risorsa per il cluster database.

> **Important**
>
> `cluster-resource-id` è diverso dall'identificatore del cluster. Per trovare un ID di risorsa del cluster in AWS Management Console Neptune, cerca nella sezione Configurazione il cluster DB in questione.

Chiavi di condizione disponibili nelle dichiarazioni di policy di accesso ai dati IAM di Neptune

Utilizzando chiavi di condizione, è possibile specificare le condizioni in una dichiarazione di policy IAM in modo che la dichiarazione diventi effettiva solo quando le condizioni sono vere.

Le chiavi di condizione che è possibile utilizzare nelle dichiarazioni di policy di accesso ai dati di Neptune rientrano nelle seguenti categorie:

- Chiavi di condizione globali: il sottoinsieme di chiavi di condizione AWS globali supportate da Neptune nelle dichiarazioni sulle politiche di accesso ai dati è elencato di seguito.
- Chiavi di condizione specifiche del servizio: si tratta di chiavi definite da Neptune specificamente per l'uso nelle dichiarazioni di policy di accesso ai dati. Al momento ce n'è solo una, neptune-db:QueryLanguage, che concede l'accesso solo se viene utilizzato un linguaggio di interrogazione specifico.

AWS chiavi di contesto delle condizioni globali supportate da Neptune nelle dichiarazioni sulle politiche di accesso ai dati

La tabella seguente elenca il sottoinsieme di chiavi di contesto di condizione globali AWS supportate da Amazon Neptune per l'uso nelle dichiarazioni di policy di accesso ai dati:

Chiavi di condizione globali che è possibile utilizzare nelle dichiarazioni di policy di accesso ai dati

Chiavi di condizione	Descrizione	Type
`aws:CurrentTime`	Filtra l'accesso in base alla data e all'ora correnti della richiesta.	`String`

Chiavi di condizione	Descrizione	Type
aws:EpochTime	Filtra l'accesso in base alla data e all'ora della richiesta espresse come valore Unix epoch.	Numeric
aws:PrincipalAccount	Filtra l'accesso in base all'account a cui appartiene il principale che effettua la richiesta.	String
aws:PrincipalArn	Filtra l'accesso in base all'ARN del principale che ha effettuato la richiesta.	String
aws:PrincipalIsAWSService	Consente l'accesso solo se la chiamata viene effettuata direttamente da un responsabile del servizio. AWS	Boolean
aws:PrincipalOrgID	Filtra l'accesso in base all'identificatore dell'organizzazione in AWS Organizations a cui appartiene il principale richiedente.	String
aws:PrincipalOrgPaths	Filtra l'accesso in base al percorso AWS Organizations per il principale che effettua la richiesta.	String
aws:PrincipalTag	Filtra l'accesso in base a un tag collegato al principale che effettua la richiesta.	String
aws:PrincipalType	Filtra l'accesso in base al tipo di principale che effettua la richiesta.	String
aws:RequestedRegion	Filtra l'accesso in base alla AWS regione chiamata nella richiesta.	String
aws:SecureTransport	Consente l'accesso solo se la richiesta è stata inviata tramite SSL.	Boolean
aws:SourceIp	Filtra l'accesso in base all'indirizzo IP del richiedente.	String

Chiavi di condizione	Descrizione	Type
aws:TokenIssueTime	Filtra l'accesso in base alla data e all'ora in cui sono state generate le credenziali di sicurezza provvisorie.	String
aws:UserAgent	Filtra l'accesso dall'applicazione client del richiedente.	String
aws:userid	Filtra l'accesso in base all'identificatore del principale del richiedente.	String
aws:ViaAWSService	Consente l'accesso solo se un AWS servizio ha effettuato la richiesta per tuo conto.	Boolean

Chiavi di condizione specifiche del servizio Neptune

Neptune supporta la seguente chiave di condizione specifica del servizio per le policy IAM:

Chiavi di condizione specifiche del servizio Neptune

Chiavi di condizione	Descrizione	Type
neptune-db:QueryLanguage	Filtra l'accesso ai dati in base al linguaggio di query utilizzato. I valori validi sonoGremlin, OpenCypher e Sparql. Le azioni supportate sono ReadDataViaQuery , WriteDataViaQuery , DeleteDataViaQuery , GetQueryStatus e CancelQuery .	String

Esempi di policy di accesso ai dati IAM di Neptune

Gli esempi seguenti mostrano come creare policy IAM personalizzate che utilizzano il controllo granulare degli accessi delle API e delle azioni del piano dati, introdotte nel rilascio 1.2.0.0 del motore Neptune.

Esempio di policy che consente l'accesso illimitato ai dati in un cluster database Neptune

La policy contenuta nell'esempio seguente consente a un utente IAM di connettersi al cluster database Neptune utilizzando l'autenticazione database IAM e utilizza il carattere "*" per trovare una corrispondenza con tutte le azioni disponibili.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

L'esempio precedente include un ARN della risorsa in un formato particolare per l'autenticazione IAM di Neptune. Per creare l'ARN, consulta Specificare le risorse dati. Nota che l'ARN utilizzato per un'autorizzazione IAM `Resource` non è lo stesso ARN assegnato al cluster in fase di creazione.

Esempio di policy che consente l'accesso in sola lettura a un cluster database Neptune

La seguente policy concede l'autorizzazione per l'accesso completo in sola lettura ai dati in un cluster database Neptune:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:Read*",
        "neptune-db:Get*",
        "neptune-db:List*"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Esempio di policy che nega l'accesso a un cluster database Neptune

L'azione IAM predefinita è negare l'accesso a un cluster database a meno che non venga concesso un effetto `Allow`. Tuttavia, la seguente politica nega tutti gli accessi a un cluster DB per un AWS account e una regione particolari, il che ha quindi la precedenza su qualsiasi `Allow` effetto.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Esempio di policy che concede l'accesso in lettura tramite query

La seguente policy concede l'autorizzazione per leggere da un cluster database Neptune solo tramite una query:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:ReadDataViaQuery",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Esempio di policy che consente solo query Gremlin

La seguente policy utilizza la chiave di condizione `neptune-db:QueryLanguage` per concedere l'autorizzazione a eseguire query su Neptune solo utilizzando il linguaggio di query Gremlin:

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "neptune-db:QueryLanguage": "Gremlin"
        }
      }
    }
  ]
}
```

Esempio di policy che consente l'accesso completo ad eccezione della gestione dei modelli Neptune ML

La seguente policy concede l'accesso completo alle operazioni del grafo Neptune, ad eccezione delle funzionalità di gestione dei modelli Neptune ML:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:CancelLoaderJob",
        "neptune-db:CancelQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:DeleteStatistics",
        "neptune-db:GetEngineStatus",
        "neptune-db:GetLoaderJobStatus",
        "neptune-db:GetQueryStatus",
        "neptune-db:GetStatisticsStatus",
        "neptune-db:GetStreamRecords",
        "neptune-db:ListLoaderJobs",
        "neptune-db:ManageStatistics",
        "neptune-db:ReadDataViaQuery",
        "neptune-db:ResetDatabase",
```

```
        "neptune-db:StartLoaderJob",
        "neptune-db:WriteDataViaQuery"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Esempio di policy che consente l'accesso alla gestione dei modelli Neptune ML

Questa policy garantisce l'accesso alle funzionalità di gestione dei modelli Neptune ML:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:CancelMLDataProcessingJob",
        "neptune-db:CancelMLModelTrainingJob",
        "neptune-db:CancelMLModelTransformJob",
        "neptune-db:CreateMLEndpoint",
        "neptune-db:DeleteMLEndpoint",
        "neptune-db:GetMLDataProcessingJobStatus",
        "neptune-db:GetMLEndpointStatus",
        "neptune-db:GetMLModelTrainingJobStatus",
        "neptune-db:GetMLModelTransformJobStatus",
        "neptune-db:ListMLDataProcessingJobs",
        "neptune-db:ListMLEndpoints",
        "neptune-db:ListMLModelTrainingJobs",
        "neptune-db:ListMLModelTransformJobs",
        "neptune-db:StartMLDataProcessingJob",
        "neptune-db:StartMLModelTrainingJob",
        "neptune-db:StartMLModelTransformJob"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Esempio di policy che concede l'accesso completo alle query

La seguente policy concede l'accesso completo alle operazioni di query del grafo Neptune, ma non a funzionalità come il ripristino rapido, i flussi, lo strumento di caricamento in blocco, la gestione dei modelli Neptune ML e così via:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:GetEngineStatus",
        "neptune-db:GetQueryStatus",
        "neptune-db:CancelQuery"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Esempio di policy che concede l'accesso completo solo alle query Gremlin

La seguente policy concede l'accesso completo alle operazioni di query del grafo Neptune utilizzando il linguaggio di query Gremlin, ma non alle query in altri linguaggi e non a funzionalità come il ripristino rapido, i flussi, lo strumento di caricamento in blocco, la gestione dei modelli Neptune ML e così via:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:GetEngineStatus",
```

```
        "neptune-db:GetQueryStatus",
        "neptune-db:CancelQuery"
      ],
      "Resource": [
        "arn:aws:neptune-db:us-east-1:123456789012:cluster-ABCD1234EFGH5678IJKL90MNOP/
*"
      ],
      "Condition": {
        "StringEquals": {
           "neptune-db:QueryLanguage":"Gremlin"
        }
      }
    }
  ]
}
```

Esempio di policy che concede l'accesso completo ad eccezione del ripristino rapido

La seguente policy concede l'accesso completo a un cluster database Neptune ad eccezione dell'utilizzo del ripristino rapido:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    },
    {
      "Effect": "Deny",
      "Action": "neptune-db:ResetDatabase",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Utilizzo di ruoli collegati ai servizi per Neptune

Amazon Neptune AWS Identity and Access Management utilizza ruoli collegati ai servizi (IAM). Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente a Neptune. I ruoli collegati ai servizi sono predefiniti da Neptune e includono tutte le autorizzazioni richieste dal servizio per chiamare altri servizi per tuo conto. AWS

> **Important**
>
> Per alcune funzionalità di gestione, Amazon Neptune utilizza la tecnologia operativa condivisa con Amazon RDS. Questo include il ruolo collegato ai servizi e le autorizzazioni API di gestione.

Un ruolo collegato ai servizi semplifica l'uso di Neptune perché non sarà più necessario aggiungere manualmente le autorizzazioni necessarie. Neptune definisce le autorizzazioni dei relativi ruoli collegati ai servizi e, salvo diversamente definito, solo Neptune potrà assumere i propri ruoli. Le autorizzazioni definite includono la policy di trust e la policy delle autorizzazioni. Una policy delle autorizzazioni specifica non può essere collegata a un'altra entità IAM.

È possibile eliminare i ruoli solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Neptune perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni sugli altri servizi che supportano i ruoli collegati ai servizi, consulta la sezione Servizi AWS che funzionano con IAM e cerca i servizi che riportano Sì nella colonna Ruolo collegato ai servizi. Scegli Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Autorizzazioni del ruolo collegato ai servizi per Neptune

Neptune utilizza `AWSServiceRoleForRDS` il ruolo collegato ai servizi per consentire a Neptune e Amazon AWS RDS di chiamare i servizi per conto delle tue istanze di database. Ai fini dell'assunzione del ruolo `AWSServiceRoleForRDS`, il ruolo collegato ai servizi `rds.amazonaws.com`considera attendibile il servizio.

La policy delle autorizzazioni del ruolo consente a Neptune di completare le seguenti azioni sulle risorse specificate:

- Operazioni su `ec2`:
  - `AssignPrivateIpAddresses`
  - `AuthorizeSecurityGroupIngress`
  - `CreateNetworkInterface`
  - `CreateSecurityGroup`
  - `DeleteNetworkInterface`
  - `DeleteSecurityGroup`
  - `DescribeAvailabilityZones`
  - `DescribeInternetGateways`
  - `DescribeSecurityGroups`
  - `DescribeSubnets`
  - `DescribeVpcAttribute`
  - `DescribeVpcs`
  - `ModifyNetworkInterfaceAttribute`
  - `RevokeSecurityGroupIngress`
  - `UnassignPrivateIpAddresses`
- Operazioni su `sns`:
  - `ListTopic`
  - `Publish`
- Operazioni su `cloudwatch`:
  - `PutMetricData`
  - `GetMetricData`
  - `CreateLogStream`
  - `PullLogEvents`
  - `DescribeLogStreams`
  - `CreateLogGroup`

Note

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi devi configurare le relative autorizzazioni. Potrebbe essere visualizzato il messaggio di errore seguente:
Unable to create the resource. (Impossibile creare la risorsa. Verify that you have permission to create service linked role. (Verifica di possedere le autorizzazioni necessarie per creare un ruolo collegato ai servizi.) Otherwise wait and try again later. (In caso contrario, attendi e riprova più tardi.
Se viene visualizzato questo messaggio, assicurati che le autorizzazioni seguenti siano abilitate:

```
{
    "Action": "iam:CreateServiceLinkedRole",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"rds.amazonaws.com"
        }
    }
}
```

Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Creazione di un ruolo collegato ai servizi per Neptune

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un'istanza o un cluster, Neptune crea il ruolo collegato ai servizi per te.

Important

Per ulteriori informazioni, consulta Comparsa di un nuovo ruolo nell'account IAM nella Guida per l'utente di IAM.

Se devi ricreare un ruolo collegato ai servizi che hai precedentemente eliminato, puoi utilizzare lo stesso processo per ricreare il ruolo nel tuo account. Quando crei un'istanza o un cluster, Neptune crea di nuovo il ruolo collegato ai servizi per te.

Modifica di un ruolo collegato ai servizi per Neptune

Neptune non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForRDS`. Dopo aver creato un ruolo collegato al servizio, non potrai modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta la sezione Modifica di un ruolo collegato ai servizi nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Neptune

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, prima di poter eliminare il ruolo collegato ai servizi associato, devi eliminare tutte le istanze e tutti i cluster.

Pulizia di un ruolo collegato ai servizi prima dell'eliminazione

Prima di utilizzare IAM per eliminare un ruolo collegato ai servizi, devi innanzitutto verificare che il ruolo non abbia sessioni attive ed eliminare tutte le risorse utilizzate dal ruolo.

Per verificare se il ruolo collegato ai servizi dispone di una sessione attiva nella console IAM

1. Accedi e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/. AWS Management Console
2. Nel pannello di navigazione della console IAM seleziona Ruoli. Quindi, scegli il nome (non la casella di controllo) del ruolo `AWSServiceRoleForRDS`.
3. Nella pagina Summary (Riepilogo) per il ruolo selezionato, scegliere la scheda Access Advisor (Consulente accessi).
4. Nella scheda Access Advisor (Consulente accessi), esamina l'attività recente per il ruolo collegato ai servizi.

> **Note**
>
> Se non si ha la certezza che Neptune stia utilizzando il ruolo `AWSServiceRoleForRDS`, è possibile provare a eliminarlo. Se il servizio sta utilizzando il ruolo, l'eliminazione non

andrà a buon fine e potrai visualizzare le regioni in cui il ruolo viene utilizzato. Se il ruolo è in uso, prima di poterlo eliminare dovrai attendere il termine della sessione. Non puoi revocare la sessione per un ruolo collegato al servizio.

Se desideri rimuovere il ruolo `AWSServiceRoleForRDS`, devi prima eliminare tutti i cluster e le istanze.

Eliminazione di tutte le istanze

Utilizza una di queste procedure per eliminare ogni istanza.

Per eliminare un'istanza (console)

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, seleziona Istanze.
3. Nell'elenco Instances (Istanze), scegliere l'istanza da eliminare.
4. Scegli Instance actions (Operazioni istanza) e quindi Delete (Elimina).
5. Se viene visualizzato il messaggio Create final Snapshot? (Creare snapshot finale?), scegliere Yes (Sì) o No.
6. Se si sceglie Yes (Sì) nella fase precedente, in Final snapshot name (Nome snapshot finale) immettere il nome dell'ultimo snapshot.
7. Scegliere Delete (Elimina).

Per eliminare un'istanza AWS CLI

Consulta `delete-db-instance` in Riferimento ai comandi AWS CLI .

Per eliminare un'istanza (API)

Per informazioni, consulta `DeleteDBInstance`.

Eliminazione di tutti i cluster

Utilizza una delle procedure seguenti per eliminare un singolo cluster, quindi ripeti la procedura per ogni cluster.

Per eliminare un cluster (console)

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nell'elenco Clusters (Cluster), scegliere il cluster da eliminare.
3. Scegliere Cluster Actions (Operazioni cluster) e successivamente Delete (Elimina).
4. Scegliere Delete (Elimina).

Per eliminare un cluster (CLI)

Consulta `delete-db-cluster` in Riferimento ai comandi AWS CLI .

Per eliminare un cluster (API)

Per informazioni, consultare `DeleteDBCluster`.

Per eliminare il ruolo collegato ai servizi `AWSServiceRoleForRDS`, puoi utilizzare la console IAM, l'interfaccia a riga di comando IAM o l'API IAM. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Autenticazione IAM con le credenziali temporanee

Amazon Neptune supporta l'autenticazione IAM con le credenziali temporanee.

Puoi utilizzare un ruolo assunto per l'autenticazione usando una policy di autenticazione IAM, come una delle policy di esempio indicate nelle sezioni precedenti.

Se usi credenziali temporanee, devi specificare `AWS_SESSION_TOKEN` oltre a `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` e `SERVICE_REGION`.

> **Note**
>
> Le credenziali temporanee scadono dopo un intervallo specificato, comprensivo del token di sessione.
> Devi aggiornare il token di sessione quando richiedi le nuove credenziali. Per ulteriori informazioni, consulta Utilizzo di credenziali di sicurezza temporanee per richiedere l'accesso alle risorse. AWS

Nelle seguenti sezioni viene descritto come consentire l'accesso e recuperare le credenziali temporanee.

Per autenticare con le credenziali temporanee

1. Creare un ruolo IAM con l'autorizzazione per accedere a un cluster Neptune. Per ulteriori informazioni sulla creazione di questo ruolo, consulta the section called “Tipi di policy IAM”.
2. Aggiungi al ruolo una relazione di trust che consenta di accedere alle credenziali.

   Recuperare le credenziali temporanee, specificando `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` e `AWS_SESSION_TOKEN`.
3. Collegarsi al cluster Neptune e firmare le richieste utilizzando le credenziali temporanee. Per ulteriori informazioni sulla connessione e sulla firma delle richieste, consulta the section called “Connessione e firma”.

Sono disponibili vari metodi per recuperare le credenziali temporanee a seconda dell'ambiente.

Argomenti

- Ottenere le credenziali temporanee utilizzando AWS CLI
- Configurazione di AWS Lambda per l'autenticazione IAM di Neptune
- Configurazione di Amazon EC2 per l'autenticazione IAM di Neptune

Ottenere le credenziali temporanee utilizzando AWS CLI

Per ottenere le credenziali utilizzando il comando AWS Command Line Interface (AWS CLI), è innanzitutto necessario aggiungere una relazione di fiducia che conceda il permesso di assumere il ruolo all' AWS utente che eseguirà il comando. AWS CLI

Aggiungere la seguente relazione di trust al ruolo di autenticazione IAM di Neptune. Se non si dispone di un ruolo di autenticazione IAM di Neptune, vedi the section called “Tipi di policy IAM”.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/test"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Per informazioni su come aggiungere la relazione di trust al ruolo, consulta Modifica della relazione di trust per un ruolo esistente nella Guida di amministrazione di AWS Directory Service .

Se la policy Neptune non è ancora collegata a un ruolo, creare un nuovo ruolo. Collegare la policy di autenticazione IAM di Neptune e aggiungere la policy di trust. Per informazioni su come creare un nuovo ruolo, consulta Creazione di un ruolo.

> **Note**
>
> Le sezioni seguenti presuppongono che tu abbia installato il AWS CLI .

Per eseguire AWS CLI manualmente

1. Digita il comando seguente per richiedere le credenziali utilizzando AWS CLI. Sostituisci l'ARN del ruolo, il nome della sessione e il profilo con i tuoi valori.

   ```
   aws sts assume-role  --role-arn  arn:aws:iam::123456789012:role/NeptuneIAMAuthRole
    --role-session-name test --profile testprofile
   ```

2. Il seguente è un esempio di output del comando. La sezione `Credentials` contiene i valori necessari.

   > **Note**
   >
   > Annota il valore `Expiration` in quanto più avanti sarà necessario per ottenere nuove credenziali.

   ```
   {
       "AssumedRoleUser": {
           "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:s3-access-example",
           "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-access-
   example"
   ```

```
    },
    "Credentials": {
        "SecretAccessKey": "9drTJvcXLB89EXAMPLELB8923FB892xMFI",
        "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/qwjzP2iEXAMPLEbw/
m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8BRi
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=",
        "Expiration": "2016-03-15T00:05:07Z",
        "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
    }
}
```

3. Imposta le variabili di ambiente usando le credenziali restituite.

```
export AWS_ACCESS_KEY_ID=ASIAJEXAMPLEXEG2JICEA
export AWS_SECRET_ACCESS_KEY=9drTJvcXLB89EXAMPLELB8923FB892xMFI
export AWS_SESSION_TOKEN=AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/qwjzP2iEXAMPLEbw/
m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8BRi
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=

export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or eu-
west-3 or eu-central-1 or me-south-1 or
                      me-central-1 or il-central-1 or af-south-1 or ap-east-1 or
 ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-southeast-2 or ap-south-1
 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

4. Collegati utilizzando uno dei seguenti metodi.

- the section called “Console Gremlin”
- the section called “Gremlin Java”
- the section called “SPARQL Java (RDF4J e Jena)”

- the section called “Esempio di Python”

Per utilizzare un script per ottenere le credenziali

1. Esegui il comando riportato qui di seguito per installare il comando jq. Lo script utilizza questo comando per analizzare l'output del AWS CLI comando.

```
sudo yum -y install jq
```

2. Crea un file denominato `credentials.sh` in un editor di testo e aggiungi il seguente testo. Sostituisci la regione del servizio, l'ARN del ruolo, il nome della sessione e il profilo con i tuoi valori.

```
#!/bin/bash

creds_json=$(aws sts assume-role  --role-arn  arn:aws:iam::123456789012:role/
NeptuneIAMAuthRole --role-session-name test --profile testprofile)

export AWS_ACCESS_KEY_ID=$(echo "$creds_json" | jq .Credentials.AccessKeyId |tr -d
 '"')
export AWS_SECRET_ACCESS_KEY=$(echo "$creds_json" |
 jq .Credentials.SecretAccessKey| tr -d '"')
export AWS_SESSION_TOKEN=$(echo "$creds_json" | jq .Credentials.SessionToken|tr -d
 '"')

export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or eu-
west-3 or eu-central-1 or me-south-1 or
                      me-central-1 or il-central-1 or af-south-1 or ap-east-1 or
 ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-southeast-2 or ap-south-1
 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

3. Collegati utilizzando uno dei seguenti metodi.

- the section called “Console Gremlin”
- the section called “Gremlin Java”
- the section called “SPARQL Java (RDF4J e Jena)”

- the section called “Esempio di Python”

Configurazione di AWS Lambda per l'autenticazione IAM di Neptune

AWS Lambda include automaticamente le credenziali ogni volta che viene eseguita la funzione Lambda.

Innanzitutto è necessario aggiungere una relazione di trust che conceda al servizio Lambda l'autorizzazione ad assumere il ruolo.

Aggiungere la seguente relazione di trust al ruolo di autenticazione IAM di Neptune. Se non si dispone di un ruolo di autenticazione IAM di Neptune, vedi the section called “Tipi di policy IAM”.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Per informazioni su come aggiungere la relazione di trust al ruolo, consulta Modifica della relazione di trust per un ruolo esistente nella Guida di amministrazione di AWS Directory Service.

Se la policy Neptune non è ancora collegata a un ruolo, creare un nuovo ruolo. Collegare la policy di autenticazione IAM di Neptune e aggiungere la policy di trust. Per ulteriori informazioni su come creare un nuovo ruolo, consulta Creazione di un nuovo ruolo nella Guida di amministrazione di AWS Directory Service .

Per accedere a Neptune da Lambda

1. Accedi AWS Management Console e apri la AWS Lambda console all'indirizzo https://console.aws.amazon.com/lambda/.
2. Creare una nuova funzione Lambda per Python versione 3.6.

3. Assegnare il ruolo `AWSLambdaVPCAccessExecutionRole` alla funzione Lambda. Questo ruolo è necessario per accedere alle risorse Neptune che sono solo VPC.
4. Assegnare il ruolo IAM di autenticazione Neptune alla funzione Lambda.

   Per ulteriori informazioni, consulta Autorizzazioni di AWS Lambda nella Guida per gli sviluppatori di AWS Lambda .
5. Copiare l'autenticazione di IAM Python di esempio nel codice di funzione Lambda.

   Per ulteriori informazioni sull'esempio e sul codice di esempio, consulta the section called “Esempio di Python”.

Configurazione di Amazon EC2 per l'autenticazione IAM di Neptune

Amazon EC2 consente di utilizzare i profili di istanza per fornire automaticamente le credenziali. Per ulteriori informazioni, consulta Utilizzo dei profili dell'istanza nella Guida per l'utente di IAM.

Innanzitutto è necessario aggiungere una relazione di trust che conceda al servizio Amazon EC2 l'autorizzazione ad assumere il ruolo.

Aggiungere la seguente relazione di trust al ruolo di autenticazione IAM di Neptune. Se non si dispone di un ruolo di autenticazione IAM di Neptune, vedi the section called “Tipi di policy IAM”.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Per informazioni su come aggiungere la relazione di trust al ruolo, consulta Modifica della relazione di trust per un ruolo esistente nella Guida di amministrazione di AWS Directory Service .

Se la policy Neptune non è ancora collegata a un ruolo, creare un nuovo ruolo. Collegare la policy di autenticazione IAM di Neptune e aggiungere la policy di trust. Per ulteriori informazioni su come creare un nuovo ruolo, consulta Creazione di un nuovo ruolo nella Guida di amministrazione di AWS Directory Service .

Per utilizzare un script per ottenere le credenziali

1. Esegui il comando riportato qui di seguito per installare il comando jq. Lo script utilizza questo comando per analizzare l'output del comando curl.

```
sudo yum -y install jq
```

2. Crea un file denominato `credentials.sh` in un editor di testo e aggiungi il seguente testo. Sostituisci la regione del servizio con il tuo valore.

```
role_name=$( curl -s http://169.254.169.254/latest/meta-data/iam/security-
credentials/ )
creds_json=$(curl -s http://169.254.169.254/latest/meta-data/iam/security-
credentials/${role_name})

export AWS_ACCESS_KEY_ID=$(echo "$creds_json" | jq .AccessKeyId |tr -d '"')
export AWS_SECRET_ACCESS_KEY=$(echo "$creds_json" | jq .SecretAccessKey| tr -d '"')
export AWS_SESSION_TOKEN=$(echo "$creds_json" | jq .Token|tr -d '"')

export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or eu-
west-3 or eu-central-1 or me-south-1 or
                      me-central-1 or il-central-1 or af-south-1 or ap-east-1 or
 ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-southeast-2 or ap-south-1
 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

3. Esegui lo script nella shell `bash` utilizzando il comando `source`:

```
source credentials.sh
```

Ancora meglio è aggiungere i comandi in questo script al file `.bashrc` sull'istanza EC2 in modo che vengano richiamati automaticamente al momento dell'accesso, rendendo disponibili le credenziali temporanee per la console di Gremlin.

4. Collegati utilizzando uno dei seguenti metodi.

   - the section called “Console Gremlin”
   - the section called “Gremlin Java”
   - the section called “SPARQL Java (RDF4J e Jena)”
   - the section called “Esempio di Python”

Registrazione e monitoraggio delle risorse Amazon Neptune

Amazon Neptune supporta vari metodi per il monitoraggio delle prestazioni e dell'utilizzo:

- Stato del cluster: controlla lo stato del motore di database a grafo di un cluster Neptune. Per ulteriori informazioni, consulta the section called “Stato dell'istanza”.
- Amazon CloudWatch — Neptune invia automaticamente i parametri e supporta anche gli CloudWatch allarmi. CloudWatch Per ulteriori informazioni, consulta the section called “Usando CloudWatch”.
- File di log di audit: puoi visualizzare, scaricare o controllare i file di log del database tramite la console. Per ulteriori informazioni, consulta the section called “Log di audit con Neptune”.
- Pubblicazione dei log su Amazon CloudWatch Logs: puoi configurare un cluster Neptune DB per pubblicare i dati dei log di controllo in un gruppo di log in Amazon Logs. CloudWatch Con CloudWatch Logs, puoi eseguire analisi in tempo reale dei dati di log, utilizzarli CloudWatch per creare allarmi e visualizzare metriche e utilizzare CloudWatch Logs per archiviare i record di log in uno storage altamente durevole. Per ulteriori informazioni, consulta Tronchi di Nettuno CloudWatch .
- AWS CloudTrail— Neptune supporta la registrazione delle API utilizzando. CloudTrail Per ulteriori informazioni, consulta the section called “Registrazione delle chiamate API Neptune con AWS CloudTrail”.
- Tagging: usa i tag per aggiungere metadati alle risorse Neptune e monitorare l'utilizzo in base a tag. Per ulteriori informazioni, consulta the section called “Applicazione di tag alle risorse Neptune”.

Convalida della conformità per Amazon Neptune

Per sapere se un Servizio AWS programma rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Scope by Compliance Program Servizi AWS e scegli il programma

di conformità che ti interessa. Per informazioni generali, consulta Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. AWS fornisce le seguenti risorse per contribuire alla conformità:

- Guide introduttive su sicurezza e conformità: queste guide all'implementazione illustrano considerazioni sull'architettura e forniscono passaggi per implementare ambienti di base incentrati sulla AWS sicurezza e la conformità.
- Progettazione per la sicurezza e la conformità HIPAA su Amazon Web Services: questo white paper descrive in che modo le aziende possono utilizzare AWS per creare applicazioni idonee all'HIPAA.

> **Note**
>
> Non Servizi AWS tutte sono idonee all'HIPAA. Per ulteriori informazioni, consulta la sezione Riferimenti sui servizi conformi ai requisiti HIPAA.

- AWS Risorse per la per la conformità: questa raccolta di cartelle di lavoro e guide potrebbe essere valida per il tuo settore e la tua località.
- AWS Guide alla conformità dei clienti: comprendi il modello di responsabilità condivisa attraverso la lente della conformità. Le guide riassumono le migliori pratiche per la protezione Servizi AWS e mappano le linee guida per i controlli di sicurezza su più framework (tra cui il National Institute of Standards and Technology (NIST), il Payment Card Industry Security Standards Council (PCI) e l'International Organization for Standardization (ISO)).
- Evaluating Resources with Rules nella AWS Config Developer Guide: il AWS Config servizio valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida e alle normative del settore.
- AWS Security Hub— Ciò Servizio AWS fornisce una visione completa dello stato di sicurezza interno. AWS La Centrale di sicurezza utilizza i controlli di sicurezza per valutare le risorse AWS e verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina Documentazione di riferimento sui controlli della Centrale di sicurezza.

- AWS Audit Manager— Ciò Servizio AWS consente di verificare continuamente AWS l'utilizzo per semplificare la gestione dei rischi e la conformità alle normative e agli standard di settore.

Resilienza in Amazon Neptune

L'infrastruttura AWS globale è costruita attorno AWS a regioni e zone di disponibilità. AWS Le regioni offrono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. Con le zone di disponibilità, è possibile progettare e gestire applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture tradizionali a data center singolo o multiplo.

Un cluster database Amazon Neptune può essere creato solo in un Amazon VPC dotato di almeno due sottoreti in almeno due zone di disponibilità. Distribuendo le istanze cluster su almeno due zone di disponibilità, Neptune contribuisce a garantire che siano disponibili istanze nel cluster database nell'improbabile caso di errore in una zona. Il volume del cluster database Neptune si estende sempre su tre zone di disponibilità per fornire spazio di archiviazione durevole con minori possibilità di perdita di dati.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, consulta Global Infrastructure.AWS

Migrazione di un grafo esistente ad Amazon Neptune

Esistono diversi strumenti e tecniche utili per eseguire la migrazione dei dati di grafi esistenti ad Amazon Neptune da un altro datastore.

Questi sono i passaggi previsti per un flusso di lavoro di migrazione semplice:

- Esporta i dati dal datastore in cui si trovano ad Amazon Simple Storage Service (Amazon S3).
- Esegui il cleanup e formatta i dati da importare.
- Caricali in un cluster database Neptune mediante lo Strumento di caricamento in blocco Neptune.
- Configura l'applicazione Gremlin o SPARQL in modo che utilizzi l'endpoint corrispondente fornito da Neptune.

> **Note**
>
> Il cluster Neptune deve essere in esecuzione in un VPC accessibile dall'applicazione.

Esistono vari modi per semplificare e automatizzare alcuni di questi passaggi, a seconda di dove sono archiviati i dati:

Argomenti

- Migrazione da Neo4j ad Amazon Neptune
- Migrazione di un grafo esistente da un server Apache TinkerPop Gremlin ad Amazon Neptune
- Migrazione di un grafo esistente da un triplestore RDF ad Amazon Neptune
- Utilizzo di AWS Database Migration Service (AWS DMS) per eseguire la migrazione da un database relazionale o NoSQL ad Amazon Neptune
- Migrazione da Blazegraph ad Amazon Neptune

Migrazione da Neo4j ad Amazon Neptune

Neo4j e Amazon Neptune sono entrambi database a grafo progettati per carichi di lavoro a grafo transazionali online e che supportano il modello di dati LPG (Labeled Property Graph, grafo di proprietà etichettate). Queste somiglianze rendono Neptune una scelta comune per i clienti che vogliono eseguire la migrazione delle applicazioni Neo4j. Tuttavia, la migrazione non si limita al rehosting poiché esistono differenze relative a supporto di linguaggi e funzionalità, caratteristiche operative, architettura dei server e capacità di archiviazione tra i due database.

Questa pagina analizza il processo di migrazione e alcuni aspetti da considerare prima di eseguire la migrazione di un'applicazione a grafo Neo4j su Neptune. Queste considerazioni si applicano in genere a qualsiasi applicazione a grafo Neo4j, indipendentemente dal fatto che si basi su un database Community, Enterprise o Aura. Sebbene ogni soluzione sia specifica e possa prevedere procedure aggiuntive, tutte le migrazioni seguono lo stesso schema generale.

Ciascuno dei passaggi descritti nelle sezioni seguenti include considerazioni e suggerimenti per semplificare il processo di migrazione. Inoltre, sono disponibili strumenti open source e post dei blog che descrivono il processo, nonché una sezione sulla compatibilità delle funzionalità con le opzioni consigliate per l'architettura.

Argomenti

Informazioni generali sulla migrazione da Neo4j a Neptune

Grazie al supporto di Neptune per il linguaggio di query openCypher, puoi spostare la maggior parte dei carichi di lavoro Neo4j che utilizzano il protocollo Bolt o HTTPS su Neptune. Tuttavia, openCypher è una specifica open source che contiene molte, ma non tutte, le funzionalità supportate da altri database come Neo4j.

Nonostante sia compatibile in molti modi, Neptune non sostituisce in tutto e per tutto Neo4j. Neptune è un servizio di database a grafo completamente gestito con funzionalità aziendali come l'alta disponibilità e l'elevata durabilità, ma la sua architettura è diversa da quella di Neo4j. Neptune si basa sulle istanze: ha una singola istanza di scrittura principale e fino a 15 istanze di replica di lettura che consentono di dimensionare orizzontalmente la capacità di lettura. Utilizzando Neptune Serverless, puoi aumentare o ridurre automaticamente la capacità di calcolo a seconda del volume di query. Ciò non dipende dall'archiviazione di Neptune, che viene ridimensionata automaticamente con l'aggiunta di dati.

Neptune supporta la specifica standard openCypher open source, versione 9. In AWS, crediamo che l'open source sia un bene per tutti e ci impegniamo a offrirne il valore ai nostri clienti, nonché a fornire l'eccellenza operativa di AWS alle community open source.

Tuttavia, molte applicazioni in esecuzione su Neo4j utilizzano anche funzionalità proprietarie che non sono open source e che Neptune non supporta. Ad esempio, Neptune non supporta le procedure APOC, alcune clausole e funzioni specifiche di Cypher e i tipi di dati `Char`, `Date` o `Duration`. Neptune trasmette automaticamente i tipi di dati mancanti utilizzando i tipi di dati supportati.

Oltre a openCypher, Neptune supporta anche il linguaggio di query Apache TinkerPop Gremlin per i grafi di proprietà (oltre a SPARQL per i dati RDF). Gremlin può interagire con openCypher sullo stesso grafo delle proprietà e, in molti casi, può essere utilizzato per fornire funzionalità che openCypher non offre. Di seguito è riportato un rapido confronto tra i due linguaggi:

	openCypher	Gremlin
Stile	Dichiarativo	Imperativo

Sintassi	Abbinamento di modelli
	`Match p=(a)-[:route]->(d)`
	`WHERE a.code='ANC'`
	`RETURN p`
		Basato su attraversamento
		`g.V().has('code', 'ANC').`
		`out('route').path().`
		`by(elementMap())`

Facilità d'uso	Ispirato a SQL, interpretabile da chi non è programmatore	Più difficile da apprender e, simile a linguaggi di programmazione come Java
Flessibilità	Bassa	Elevata

	openCypher	Gremlin
Supporto delle query	Query basate su stringhe	Query basate su stringhe o codice in linea supportato da librerie client
Client	HTTPS e Bolt	HTTPS e Websocket

In generale, non è necessario modificare il modello di dati per eseguire la migrazione da Neo4j a Neptune poiché sia Neo4j che Neptune supportano i dati LPG (Labeled Property Graph). Tuttavia, Neptune ha alcune differenze in termini di architettura e modello di dati da usare per ottimizzare le prestazioni. Ad esempio:

- Gli ID Neptune vengono trattati come elementi prioritari.
- Neptune utilizza le policy di AWS Identity and Access Management (IAM) per proteggere l'accesso ai dati del grafo in modo flessibile e dettagliato.
- Neptune offre diversi modi per utilizzare i notebook Jupyter al fine di eseguire query e visualizzare i risultati. Neptune funziona anche con strumenti di visualizzazione di terze parti.
- >Neptune non dispone di un sostituto immediato della libreria Graph Data Science (GDS) di Neo4j, Neptune ma al momento supporta l'analisi dei grafi mediante varie soluzioni. Ad esempio, diversi notebook di esempio dimostrano come sfruttare l'integrazione di Neptune con l'SDK Pandas di AWS all'interno degli ambienti Python per eseguire analisi sui dati dei grafi.

Contatta l'assistenza AWS o il team del tuo account AWS se hai altre domande. Utilizziamo il tuo feedback per dare priorità a nuove funzionalità in grado di soddisfare le tue esigenze.

Preparazione per la migrazione da Neo4j a Neptune

Approcci alla migrazione

Quando si esegue la migrazione di un'applicazione Neo4j su Neptune, consigliamo di seguire una di due strategie: quella di re-platforming o quella di refactoring/re-architecting. Per ulteriori informazioni sulle strategie di migrazione, consulta 6 Strategies for Migrating Applications to the Cloud, un post del blog di Stephen Orban.

L'approccio di re-platforming, a volte chiamato lift-tinker-and-shift, prevede i seguenti passaggi:

- Identificazione dei casi d'uso che l'applicazione deve soddisfare.
- Modifica del modello di dati a grafo e dell'architettura applicativa esistenti per soddisfare al meglio le esigenze di carico di lavoro utilizzando le funzionalità di Neptune.
- Definizione della migrazione di dati, query e altre parti dell'applicazione di origine nel modello e nell'architettura di destinazione.

Questo approccio "a ritroso" consente di eseguire la migrazione dell'applicazione nel tipo di soluzione Neptune che potresti progettare se si trattasse di un progetto completamente nuovo.

L'approccio del refactoring, al contrario, prevede questi passaggi:

- Identificazione dei componenti dell'implementazione esistente, tra cui infrastruttura, dati, query e funzionalità dell'applicazione.
- Individuazione di soluzioni Neptune equivalenti che possano essere usate per creare un'implementazione comparabile.

Questo approccio "progressivo" punta a sostituire un'implementazione con un'altra.

Nella pratica, è probabile che adotterai un mix di questi due approcci. Potresti iniziare con un caso d'uso, ad esempio progettare l'architettura Neptune di destinazione, per poi passare all'implementazione Neo4j esistente per identificare i vincoli e gli elementi fondamentali che dovrai rispettare. Ad esempio, potresti dover continuare a garantire l'integrazione con altri sistemi esterni o a offrire API specifiche ai consumatori della tua applicazione a grafo. Con queste informazioni, è possibile determinare se esistono già dei dati da trasferire nel modello di destinazione e quali devono provenire da altre origini.

In altre fasi, potresti iniziare analizzando una parte specifica dell'implementazione Neo4j come fonte migliore di informazioni su ciò che deve fare l'applicazione. Questo tipo di analisi storica dell'applicazione esistente è utile per definire un caso d'uso mirato a utilizzare le funzionalità di Neptune.

Sia che tu stia creando una nuova applicazione utilizzando Neptune o eseguendo la migrazione di un'applicazione esistente da Neo4j, ti consigliamo di lavorare a ritroso partendo dai casi d'uso per progettare un modello di dati, una serie di query e un'architettura applicativa che soddisfino le esigenze aziendali.

Differenze architettoniche tra Neptune e Neo4j

Quando i clienti prendono in considerazione per la prima volta la migrazione di un'applicazione da Neo4j a Neptune, sono spesso tentati di eseguire un confronto di somiglianza sulla base della dimensione dell'istanza. Tuttavia, le architetture di Neo4j e Neptune hanno differenze fondamentali. Neo4j si basa su un approccio all-in-one in cui il caricamento dei dati, l'ETL dei dati, le query delle applicazioni, l'archiviazione di dati e le operazioni di gestione avvengono tutti nello stesso set di risorse di calcolo, come le istanze EC2.

Neptune, al contrario, è un database a grafo incentrato su OLTP in cui l'architettura separa le responsabilità e le risorse sono separate in modo che possano scalare in modo dinamico e indipendente.

Durante la migrazione da Neo4j a Neptune, devi determinare i requisiti di durabilità, disponibilità e scalabilità dei dati dell'applicazione. L'architettura di cluster di Neptune semplifica la progettazione di applicazioni che richiedono durabilità, disponibilità e scalabilità elevate. Se comprendi bene l'architettura di cluster di Neptune, potrai progettare una topologia di cluster Neptune per soddisfare questi requisiti.

Architettura di cluster di Neo4j

Molte applicazioni di produzione utilizzano il clustering causale di Neo4j per garantire durabilità dei dati, alta disponibilità e scalabilità. L'architettura di clustering di Neo4j utilizza istanze di server principali e di replica di lettura:

- I server principali garantiscono la durabilità dei dati e la tolleranza agli errori mediante la replica dei dati con il protocollo Raft.
- Le repliche di lettura utilizzano l'invio dei log delle transazioni per replicare in modo asincrono i dati per carichi di lavoro con throughput di lettura elevato.

Ogni istanza di un cluster, che si tratti di un server principale o di una replica di lettura, contiene una copia completa dei dati a grafo.

Architettura di cluster di Neptune

Un cluster di Neptune è composto da un'istanza writer principale e da un massimo di 15 istanze di replica di lettura. Tutte le istanze del cluster condividono lo stesso servizio di archiviazione distribuito sottostante, separato rispetto alle istanze.

- L'istanza writer principale coordina tutte le operazioni di scrittura sul database ed è scalabile verticalmente per fornire un supporto flessibile per diversi carichi di lavoro di scrittura. Supporta anche le operazioni di lettura.
- Le istanze di replica di lettura supportano le operazioni di lettura dal volume di archiviazione sottostante e consentono la scalabilità orizzontale per supportare carichi di lavoro di lettura elevati. Garantiscono, inoltre, un'elevata disponibilità poiché fungono da destinazioni di failover per l'istanza principale.

  > **Note**
  >
  > Nel caso di carichi di lavoro con attività intensive di scrittura, è consigliabile ridimensionare le istanze di replica di lettura in modo che abbiano le stesse dimensioni dell'istanza writer, così da garantire che i reader possano rimanere in linea con le modifiche dei dati.

- Il volume di archiviazione sottostante ridimensiona automaticamente la capacità di archiviazione con l'aumento dei dati del database (fino a 128 TiB di archiviazione).

Le dimensioni delle istanze sono dinamiche e indipendenti. Ogni istanza può essere ridimensionata mentre il cluster è in esecuzione e le repliche di lettura possono essere aggiunte o rimosse durante l'esecuzione del cluster.

La funzionalità Neptune Serverless può aumentare e ridurre automaticamente la capacità di elaborazione in base all'aumento e alla diminuzione della domanda. In questo modo, è possibile non solo ridurre il sovraccarico amministrativo, ma anche configurare il database per gestire picchi di domanda elevati senza ridurre le prestazioni o richiedere un provisioning eccessivo.

È possibile arrestare un cluster di Neptune per un massimo di sette giorni.

Neptune supporta anche il dimensionamento automatico per adattare automaticamente le dimensioni delle istanze reader in base al carico di lavoro.

Utilizzando la funzionalità di database globale Neptune, puoi eseguire il mirroring di un cluster in un massimo di altre 5 regioni.

Neptune offre anche la tolleranza agli errori per impostazione predefinita:

- Il volume del cluster che fornisce l'archiviazione di dati a tutte le istanze si estende su più zone di disponibilità (AZ) di una singola Regione AWS. Ogni AZ contiene una copia completa dei dati del cluster.
- Se l'istanza principale non è più disponibile, Neptune esegue automaticamente il failover su una replica di lettura esistente senza alcuna perdita di dati, in genere in meno di 30 secondi. Se nel cluster non esistono repliche di lettura, Neptune effettua automaticamente il provisioning di una nuova istanza primaria, sempre senza perdere dati.

Ciò significa che, durante la migrazione da un cluster causale Neo4j a Neptune, non è necessario progettare la topologia del cluster in modo esplicito in modo che abbia durabilità dei dati e disponibilità elevate. Puoi così stabilire le dimensioni del cluster in base ai carichi di lavoro di lettura e scrittura previsti, nonché a eventuali requisiti di maggiore disponibilità. Ecco come puoi fare:

- Per scalare le operazioni di lettura, aggiungi istanze di replica di lettura o abilita la funzionalità Neptune Serverless.
- Per migliorare la disponibilità, distribuisci l'istanza primaria e le repliche nel cluster su più zone di disponibilità (AZ).
- Per ridurre i tempi di failover, fornisci almeno un'istanza di replica di lettura che possa fungere da destinazione di failover per l'istanza primaria. Per determinare l'ordine di promozione a istanza primaria delle repliche di lettura dopo un errore, assegna una priorità a ciascuna di esse. È consigliabile assicurarsi che una destinazione di failover disponga di una classe di istanza in grado di gestire il carico di lavoro di scrittura dell'applicazione, se promossa a istanza primaria.

Differenze di archiviazione di dati tra Neptune e Neo4j

Neptune utilizza un modello di dati a grafo basato su un modello quad nativo. Durante la migrazione dei dati a Neptune, devi conoscere le molte differenze nell'architettura del modello di dati e nel livello di archiviazione per utilizzare in modo ottimale l'archiviazione condivisa distribuita e scalabile fornita da Neptune:

- Neptune non utilizza schemi o vincoli definiti in modo esplicito. Consente di aggiungere nodi, archi e proprietà in modo dinamico senza dover definire preventivamente lo schema. Neptune non limita

i valori e i tipi di dati archiviati, ad eccezione di quanto indicato nei limiti di Neptune. Come parte dell'architettura di archiviazione di Neptune, i dati vengono inoltre indicizzati automaticamente in modo da gestire molti dei pattern di accesso più comuni. Questa architettura di archiviazione elimina il sovraccarico operativo legato alla creazione e alla gestione dello schema del database e all'ottimizzazione degli indici.

- Neptune offre un'architettura di archiviazione distribuita e condivisa univoca che si ridimensiona automaticamente in blocchi da 10 GB man mano che le esigenze di archiviazione del database crescono, fino a 128 tebibyte (TiB). Questo livello di archiviazione è affidabile, duraturo e tollerante agli errori, con dati copiati 6 volte, due volte in ciascuna delle 3 zone di disponibilità. Per impostazione predefinita, tutti i cluster di Neptune hanno un livello di archiviazione di dati ad alta disponibilità e tollerante agli errori. L'architettura di archiviazione di Neptune riduce i costi ed elimina la necessità di eseguire il provisioning o l'over-provisioning dell'archiviazione per gestire la crescita futura dei dati.

Prima di eseguire la migrazione dei dati su Neptune, è bene acquisire familiarità con il modello di dati a grafo delle proprietà e la semantica delle transazioni di Neptune.

Differenze operative tra Neptune e Neo4j

Neptune è un servizio completamente gestito che automatizza molte delle normali attività operative da svolgere quando utilizzi database locali o autogestiti come Neo4j Enterprise o Community Edition:

- Backup automatici: Neptune esegue automaticamente il backup del volume del cluster e conserva il backup per un periodo specificato da te (da 1 a 35 giorni). Questi backup sono continui e incrementali, in modo da poter eseguire rapidamente un ripristino in qualsiasi punto nel periodo di conservazione. Durante la scrittura dei dati di backup, non si verifica alcun impatto sulle prestazioni o interruzione del funzionamento del servizio del database.
- Snapshot manuali: Neptune consente di creare uno snapshot del volume di archiviazione del cluster database per eseguire il backup dell'intero cluster database. Questo tipo di snapshot può quindi essere utilizzato per ripristinare il database, crearne una copia e condividerlo tra più account.
- Clonazione: Neptune supporta una funzionalità di clonazione per creare rapidamente cloni di un database a costi contenuti. I cloni utilizzano un protocollo copy-on-write che richiede solo uno spazio aggiuntivo minimo dopo la creazione. La clonazione del database è un modo efficace per provare nuove funzionalità o aggiornamenti di Neptune senza interrompere il cluster di origine.
- Monitoraggio: Neptune offre vari metodi per monitorare le prestazioni e l'utilizzo del cluster, tra cui:
  - Stato dell'istanza

  - Integrazione con Amazon CloudWatch e AWS CloudTrail
  - Funzionalità di log di audit
  - Notifiche degli eventi
  - Assegnazione di tag
- Sicurezza: per impostazione predefinita, Neptune fornisce un ambiente sicuro. Un cluster risiede all'interno di un VPC privato che garantisce l'isolamento della rete da altre risorse. Tutto il traffico è crittografato tramite SSL e tutti i dati inattivi vengono crittografati utilizzando AWS KMS.

  Inoltre, Neptune si integra con AWS Identity and Access Management (IAM) per fornire l'autenticazione. Se specifichi chiavi delle condizioni IAM, puoi utilizzare le policy IAM per offrire un controllo granulare degli accessi per le azioni sui dati.

Differenze di strumenti e integrazioni tra Neptune e Neo4j

Neptune ha un'architettura diversa per le integrazioni e gli strumenti rispetto a Neo4j, il che può influire sull'architettura dell'applicazione. Neptune utilizza le risorse di calcolo del cluster per elaborare le query, ma sfrutta altri servizi AWS all'avanguardia per funzionalità come la ricerca full-text (con OpenSearch), l'ETL (con Glue) e così via. Per un elenco completo di queste integrazioni, vedi Integrazioni di Neptune.

Provisioning dell'infrastruttura durante la migrazione da Neo4j a Neptune

I cluster Amazon Neptune sono progettati per essere dimensionanti in base a tre aspetti: archiviazione, capacità di scrittura e capacità di lettura. Le sezioni seguenti illustrano le opzioni specifiche da considerare durante la migrazione.

Provisioning dell'archiviazione

L'archiviazione per qualsiasi cluster Neptune viene fornita automaticamente, senza che tu debba effettuare alcuna attività amministrativa. Si ridimensiona dinamicamente in blocchi da 10 GB man mano che aumentano le esigenze di archiviazione del cluster. Di conseguenza, non è necessario stimare ed eseguire il provisioning o l'over-provisioning dell'archiviazione per gestire la crescita futura dei dati.

Provisioning della capacità di scrittura

Neptune fornisce una singola istanza di scrittura che può essere dimensionata verticalmente fino a qualsiasi dimensione di istanza disponibile nella pagina dei prezzi di Neptune. Durante la lettura e la scrittura dei dati su un'istanza writer, tutte le transazioni sono conformi alla normativa ACID, secondo l'isolamento dei dati definito in Livelli di isolamento della transazione in Neptune.

La scelta della dimensione ottimale per un'istanza writer richiede l'esecuzione di test di carico per determinare la dimensione ottimale dell'istanza per il carico di lavoro. Qualsiasi istanza all'interno di Neptune può essere ridimensionata in qualsiasi momento modificando la classe dell'istanza database. È possibile stimare la dimensione iniziale dell'istanza in base alla simultaneità e alla latenza media delle query, come descritto di seguito in Stima delle dimensioni ottimali delle istanze durante il provisioning del cluster.

Provisioning della capacità di lettura

Neptune è progettato in modo da dimensionare le istanze di replica di lettura sia orizzontalmente, aggiungendone fino a 15 all'interno di un cluster (o anche un numero maggiore in un database globale Neptune), sia verticalmente fino a qualsiasi dimensione di istanza disponibile nella pagina dei prezzi di Neptune. Tutte le istanze di replica di lettura Neptune utilizzano lo stesso volume di archiviazione sottostante, consentendo una replica trasparente dei dati con ritardi minimi.

Oltre a consentire il dimensionata orizzontale delle richieste di lettura all'interno di un cluster Neptune, le repliche di lettura fungono anche da destinazioni di failover per l'istanza writer, così da garantire l'elevata disponibilità. Consulta Linee guida operative di base per Amazon Neptune per suggerimenti su come determinare il numero e il posizionamento appropriati delle repliche di lettura nel cluster.

Per le applicazioni in cui la connettività e il carico di lavoro sono imprevedibili, Neptune supporta anche una funzionalità di dimensionamento automatico che può regolare automaticamente il numero di repliche Neptune in base ai criteri che specifichi.

Per determinare la dimensione e il numero ottimali di istanze di replica di lettura è necessario eseguire test di carico per stabilire le caratteristiche del carico di lavoro di lettura da supportare. Qualsiasi istanza all'interno di Neptune può essere ridimensionata in qualsiasi momento modificando la classe dell'istanza database. È possibile stimare la dimensione iniziale dell'istanza in base alla simultaneità e alla latenza media delle query, come descritto nella sezione successiva.

Utilizzo di Neptune Serverless per ridimensionare automaticamente le istanze di lettura e scrittura in base alle esigenze

Sebbene sia spesso utile poter stimare la capacità di elaborazione richiesta dai carichi di lavoro previsti, è possibile configurare la funzionalità Neptune Serverless per aumentare e ridurre automaticamente la capacità di lettura e scrittura. Ciò è utile per far fronte ai requisiti di picco e, allo stesso tempo, per ridurre automaticamente la capacità quando la domanda diminuisce.

Stima delle dimensioni ottimali delle istanze durante il provisioning del cluster

Per stimare le dimensioni ottimali delle istanze, occorre conoscere la latenza media delle query in Neptune, il momento in cui è in esecuzione il carico di lavoro e il numero di query simultanee elaborate. Una stima approssimativa della dimensione dell'istanza può essere calcolata come la latenza media delle query moltiplicata per il numero di query simultanee. In questo modo si ottiene il numero medio di thread simultanei necessari per gestire il carico di lavoro.

Ogni vCPU in un'istanza Neptune può supportare due thread di query simultanei. Di conseguenza, dividendo i thread per 2, si ottiene il numero di vCPU richiesto, che può quindi essere correlato alla dimensione appropriata dell'istanza nella pagina dei prezzi di Neptune. Ad esempio:

```
Average Query Latency:         30ms (0.03s)
Number of concurrent queries:  1000/second

Number of threads needed:      0.03 x 1000 = 30 threads
Number of vCPUs needed:        30 / 2 = 15 vCPUs
```

Correlando quanto sopra al numero di vCPU in un'istanza, otteniamo una stima approssimativa secondo la quale `r5.4xlarge` sarebbe l'istanza sarebbe consigliata per questo carico di lavoro. Questa stima è approssimativa e serve solo a fornire una guida iniziale sulla selezione della

dimensione dell'istanza. Qualsiasi applicazione deve essere sottoposta ad attività per trovare le dimensioni corrette al fine di determinare il numero e il tipo di istanze appropriati per il carico di lavoro.

È inoltre necessario tenere conto dei requisiti di memoria e di elaborazione. Neptune offre le prestazioni migliori se i dati a cui si accede tramite query sono disponibili nella cache del pool di buffer della memoria principale. Il provisioning di una quantità sufficiente di memoria può anche ridurre in modo significativo i costi di I/O.

Ulteriori dettagli e indicazioni sul dimensionamento delle istanze in un cluster Neptune sono disponibili nella pagina Dimensionamento delle istanze database in un cluster database Neptune.

Migrazione dei dati da Neo4j a Neptune

Quando si esegue una migrazione da Neo4j ad Amazon Neptune, la migrazione dei dati è una fase importante del processo. Esistono diversi approcci per eseguire la migrazione dei dati. Quello corretto è determinato dalle esigenze dell'applicazione, dalla dimensione dei dati e dal tipo di migrazione desiderata. Tuttavia, per tutte queste migrazioni è necessario prendere in considerazione gli stessi aspetti, alcuni dei quali sono descritti di seguito.

> **Note**
>
> Consulta l'articolo sulla migrazione di un database a grafo da Neo4j a Neptune con un'utilità completamente automatizzata nel AWSblog sui database per una guida dettagliata e completa di un esempio per eseguire una migrazione di dati offline.

Valutazione della migrazione dei dati da Neo4j a Neptune

Il primo passo per valutare una migrazione dei dati è determinare come eseguirla. Le opzioni dipendono dall'architettura dell'applicazione di cui eseguire la migrazione, dalla dimensione dei dati e dalle esigenze di disponibilità durante la migrazione. In generale, le migrazioni rientrano in due categorie: online o offline.

Le migrazioni offline di solito sono le più semplici da eseguire, dato che l'applicazione non accetta traffico di lettura o scrittura durante la migrazione. Quando l'applicazione smette di accettare traffico, i dati possono essere esportati, ottimizzati e importati, quindi l'applicazione testata prima di essere abilitata di nuovo.

Le migrazioni online sono più complesse perché l'applicazione deve comunque accettare traffico di lettura e scrittura durante la migrazione dei dati. Le esigenze esatte di ogni migrazione online possono essere diverse, ma l'architettura generale potrebbe essere simile a questa:

- Un feed di modifiche continue al database deve essere abilitato in Neo4j configurando Neo4j Streams come origine per un cluster Kafka.
- Una volta completata questa operazione, è possibile eseguire un'esportazione del sistema in esecuzione secondo le istruzioni riportate in Esportazione dei dati da Neo4j durante la migrazione a Neptune, oltre ad annotare l'ora indicata per una successiva correlazione con l'argomento Kafka.
- I dati esportati vengono quindi importati in Neptune secondo le istruzioni riportate in Importazione dei dati da Neo4j durante la migrazione a Neptune.

- I dati modificati dal flusso Kafka possono quindi essere copiati nel cluster Neptune utilizzando un'architettura simile a quella descritta in Writing to Amazon Neptune from Amazon Kinesis Data Streams. Tieni presente che la replica delle modifiche può essere eseguita in parallelo per convalidare la nuova architettura e le prestazioni dell'applicazione.
- Dopo la convalida della migrazione dei dati, il traffico dell'applicazione può essere reindirizzato al cluster Neptune e l'istanza Neo4j può essere disattivata.

Ottimizzazioni del modello di dati per la migrazione da Neo4j a Neptune

Sia Neptune che Neo4j supportano i modelli di dati LPG (Labeled Property Graph, grafo di proprietà etichettate). Tuttavia, Neptune ha alcune differenze in termini di architettura e modello di dati da usare per ottimizzare le prestazioni:

Ottimizzazione degli ID di nodi ed archi

Neo4j genera automaticamente ID numerici lunghi. Usando Cypher puoi fare riferimento ai nodi per ID, ma questa procedura è generalmente sconsigliata a favore della ricerca dei nodi in base a proprietà indicizzate.

Neptune consente di fornire ID personalizzati basati su stringhe per vertici e archi. Se non fornisci i tuoi ID, Neptune genera automaticamente rappresentazioni di stringa degli UUID per nuovi archi e vertici.

Se esegui la migrazione dei dati da Neo4j a Neptune esportandoli da Neo4j e poi importandoli in blocco in Neptune, puoi conservare gli ID di Neo4j. I valori numerici generati da Neo4j possono fungere da ID forniti dall'utente durante l'importazione in Neptune, dove sono rappresentati come stringhe anziché valori numerici.

Tuttavia, in alcuni casi potresti voler promuovere una proprietà di vertice in modo che diventi un ID di vertice. Proprio come la ricerca di un nodo mediante una proprietà indicizzata è il modo più veloce per trovare un nodo in Neo4j, la ricerca di un vertice per ID è il modo più veloce per trovare un vertice in Neptune. Pertanto, se riesci a identificare una proprietà di vertice adatta che contenga valori univoci, dovresti prendere in considerazione la possibilità di sostituire il vertice ~id con il valore della proprietà denominato nei file CSV caricati in blocco. Se lo fai, dovrai anche riscrivere tutti i valori ~from e ~to degli archi corrispondenti nei file CSV.

Vincoli di schema durante la migrazione dei dati da Neo4j a Neptune

All'interno di Neptune, l'unico vincolo di schema disponibile è l'unicità dell'ID di un nodo o un arco. Le applicazioni che devono sfruttare un vincolo di unicità dovrebbero prendere in considerazione questo

approccio per ottenere un vincolo di unicità specificando l'ID del nodo o dell'arco. Se l'applicazione utilizza più colonne come vincolo di unicità, l'ID può essere impostato su una combinazione di questi valori. Ad esempio, `id=123, code='SEA'` potrebbe essere rappresentato come`ID='123_SEA'` per ottenere un vincolo di unicità complesso.

Ottimizzazione della direzione degli archi durante la migrazione dei dati da Neo4j a Neptune

Quando nodi, archi o proprietà vengono aggiunti a Neptune, vengono indicizzati automaticamente in tre modi diversi, con un quarto indice facoltativo. Grazie al modo in cui Neptune crea e utilizza gli indici, le query che seguono gli archi in uscita sono più efficienti di quelle che utilizzano gli archi in entrata. Per quanto riguarda il modello di archiviazione di dati a grafo di Neptune, si tratta di ricerche basate su argomenti che utilizzano l'indice SPOG.

Se durante la migrazione del modello di dati e delle query su Neptune scopri che le query più importanti si basano sull'attraversamento degli archi in entrata con un alto livello di dispersione, potrebbe essere utile modificare il modello in modo che questi attraversamenti seguano invece gli archi in uscita, specialmente se non puoi specificare le etichette che gli archi devono attraversare. A tale scopo, inverti la direzione degli archi pertinenti e aggiorna le etichette degli archi in modo che riflettano la semantica di questo cambio di direzione. Ad esempio, potresti cambiare:

```
person_A – parent_of – person_B
   to:
person_B – child_of – person_A
```

Per apportare questa modifica in un file CSV caricato in blocco con gli archi, basta scambiare le intestazioni delle colonne `~from` e `~to` e aggiornare i valori della colonna `~label`.

In alternativa all'inversione della direzione degli archi, puoi abilitare un quarto indice di Neptune, ovvero l'indice OSGP, che rende molto più efficiente l'attraversamento degli archi in entrata o le ricerche basate su oggetti. Tuttavia, l'abilitazione di questo quarto indice ridurrà le frequenze di inserimento e richiederà più archiviazione.

Ottimizzazione dei filtri durante la migrazione dei dati da Neo4j a Neptune

Neptune è ottimizzato per funzionare al meglio quando le proprietà vengono filtrate in base alla proprietà più selettiva disponibile. Quando si utilizzano più filtri, viene trovato il set di elementi corrispondenti per ciascuno e poi viene calcolata la sovrapposizione di tutti i set corrispondenti. Se possibile, la combinazione di più proprietà in un'unica proprietà riduce al minimo il numero di ricerche nell'indice e diminuisce la latenza di una query.

Ad esempio, questa query utilizza due ricerche nell'indice e un join:

```
MATCH (n) WHERE n.first_name='John' AND n.last_name='Doe' RETURN n
```

Questa query recupera le stesse informazioni utilizzando una singola ricerca nell'indice:

```
MATCH (n) WHERE n.name='John Doe' RETURN n
```

Neptune supporta tipi di dati diversi rispetto a Neo4j.

Mappature dei tipi di dati Neo4j con i tipi di dati supportati da Neptune

- Logico: `Boolean`

  Da mappare in Neptune con `Bool` o `Boolean`.
- Numerico: `Number`

  Da mappare in Neptune con il più stretto dei seguenti tipi di openCypher di Neptune che possono supportare tutti i valori della proprietà numerica in questione:

  ```
  Byte
  Short
  Integer
  Long
  Float
  Double
  ```
- Testo: `String`

  Da mappare in Neptune con `String`.
- Punto temporale:

  ```
  Date
  Time
  LocalTime
  DateTime
  LocalDateTime
  ```

  Da mappare in Neptune con `Date` come UTC, usando uno dei seguenti formati ISO-8601 supportati da Neptune:

```
yyyy-MM-dd
yyyy-MM-ddTHH:mm
yyyy-MM-ddTHH:mm:ss
yyyy-MM-ddTHH:mm:ssZ
```

- Durata temporale: `Duration`

  Da mappare in Neptune con un valore numerico per l'aritmetica delle date, se necessario.

- Spaziale: `Point`

  Da mappare in Neptune con valori numerici dei componenti, ognuno dei quali diventa quindi una proprietà separata. In alternativa, esprimilo come valore di stringa da interpretare dall'applicazione client. Tieni presente che l'integrazione della ricerca full-text di Neptune con OpenSearch consente di indicizzare le proprietà di geolocalizzazione.

Migrazione di proprietà multivalore da Neo4j a Neptune

Neo4j consente di archiviare elenchi omogenei di tipi semplici come proprietà sia dei nodi che degli archi. Questi elenchi possono contenere valori duplicati.

Neptune, tuttavia, consente solo la cardinalità singola o degli insiemi per le proprietà dei vertici e la cardinalità singola per le proprietà degli archi nei dati a grafo delle proprietà. Di conseguenza, non è possibile eseguire la migrazione diretta delle proprietà dell'elenco dei nodi Neo4j che contengono valori duplicati nelle proprietà di vertici Neptune o delle proprietà dell'elenco di relazioni Neo4j nelle proprietà degli archi Neptune.

Ecco alcune possibili strategie per la migrazione delle proprietà dei nodi multivalore Neo4j con valori duplicati in Neptune:

- Scarta i valori duplicati e converti la proprietà multivalore dei nodi Neo4j in una proprietà di vertice Neptune con cardinalità degli insiemi. Tieni presente che l'insieme Neptune potrebbe non riflettere l'ordine degli elementi nella proprietà multivalore Neo4j originale.
- Converti la proprietà multivalore del nodo Neo4j in una rappresentazione di stringa di un elenco in formato JSON in una proprietà di stringa di vertice Neptune.
- Estrai ogni valore della proprietà multivalore in un vertice separato con una proprietà di valore e collega i vertici a quello principale utilizzando un arco etichettato con il nome della proprietà.

Ecco alcune possibili strategie per la migrazione delle proprietà delle relazioni multivalore Neo4j in Neptune:

- Converti la proprietà della relazione multivalore Neo4j in una rappresentazione di stringa di un elenco in formato JSON e archivialo come proprietà di stringa di arco Neptune.
- Rifattorizza la relazione Neo4j negli archi Neptune in entrata e in uscita collegati a un vertice intermedio. Estrai ogni valore della proprietà della relazione multivalore in un vertice separato con una proprietà di valore, poi estrai i vertici nel vertice intermedio utilizzando un arco etichettato con il nome della proprietà.

Tieni presente che una rappresentazione in formato di stringa di un elenco in formato JSON è opaca per il linguaggio di query openCypher, sebbene questo includa un predicato `CONTAINS` che consente ricerche semplici all'interno di valori di stringa.

Esportazione dei dati da Neo4j durante la migrazione a Neptune

Al momento di esportare i dati da Neo4j, utilizza le procedure APOC per eseguire l'esportazione in CSV o GraphML. Sebbene sia possibile esportare i dati in altri formati, esistono strumenti open source per convertire i dati CSV esportati da Neo4j nel formato di caricamento in blocco Neptune, oltre a strumenti open source per convertire i dati GraphML esportati da Neo4j nel formato di caricamento in blocco Neptune.

Puoi anche esportare i dati direttamente in Amazon S3 utilizzando le varie procedure APOC. L'esportazione in un bucket Amazon S3 è disabilitata per impostazione predefinita, ma può essere abilitata utilizzando le procedure relative all'esportazione in Amazon S3, descritte nella documentazione APOC di Neo4j.

Importazione dei dati da Neo4j durante la migrazione a Neptune

Per importare i dati in Neptune, puoi utilizzare lo strumento di caricamento in blocco Neptune o la logica dell'applicazione in un linguaggio di query supportato come openCypher.

L'uso dello strumento di caricamento in blocco Neptune è l'approccio preferito per importare grandi quantità di dati perché ottimizza la resa dell'importazione se si seguono le best practice del caso. Lo strumento di caricamento in blocco supporta due diversi formati CSV in cui è possibile convertire i dati esportati da Neo4j utilizzando le utilità open source menzionate in precedenza nella sezione Esportazione dei dati.

Puoi anche usare openCypher per importare i dati con una logica personalizzata per l'analisi, la trasformazione e l'importazione. È possibile inviare le query openCypher tramite l'endpoint HTTPS (scelta consigliata) o utilizzando il driver Bolt.

Migrazione dei dati da Neo4j a Neptune

Dopo aver eseguito la migrazione dei dati da Neo4j a Neptune, il passaggio successivo prevede la migrazione dell'applicazione stessa. Come per i dati, esistono diversi approcci alla migrazione dell'applicazione in base agli strumenti utilizzati, ai requisiti, alle differenze architettoniche e così via. Di seguito sono descritti gli aspetti che di solito è necessario prendere in considerazione durante questo processo.

Migrazione delle connessioni durante il passaggio da Neo4j a Neptune

Se attualmente non utilizzi i driver Bolt o preferisci utilizzare un'alternativa, puoi connetterti all'endpoint HTTPS che fornisce l'accesso completo ai dati restituiti.

Se disponi di un'applicazione che utilizza il protocollo Bolt, puoi eseguire la migrazione delle connessioni a Neptune e consentire alle applicazioni di connettersi utilizzando gli stessi driver che utilizzavi in Neo4j. Per connetterti a Neptune, potrebbe essere necessario apportare una o più modifiche all'applicazione, come descritto di seguito:

- L'URL e la porta dovranno essere aggiornati in modo da utilizzare gli endpoint e la porta del cluster (quella predefinita è 8182).
- Neptune richiede che tutte le connessioni utilizzino il protocollo SSL, quindi devi specificare che ogni connessione è crittografata.
- Neptune gestisce l'autenticazione tramite l'assegnazione di policy e ruoli IAM. Le policy e i ruoli IAM forniscono un livello estremamente flessibile di gestione degli utenti all'interno dell'applicazione, quindi è importante leggere e comprendere le informazioni contenute nella panoramica di IAM prima di configurare il cluster.
- Le connessioni Bolt si comportano diversamente in Neptune rispetto a Neo4j, come spiegato in Comportamento della connessione Bolt in Neptune.
- Puoi trovare ulteriori informazioni e suggerimenti in Best practice di Neptune per l'utilizzo di openCypher e Bolt.

Sono disponibili esempi di codice per linguaggi di uso comune come Java, Python, .NET e NodeJS, nonché per scenari di connessione, ad esempio l'utilizzo dell'autenticazione IAM, in Utilizzo del protocollo Bolt per effettuare query openCypher in Neptune.

Instradamento delle query verso le istanze del cluster durante il passaggio da Neo4j a Neptune

Le applicazioni client Neo4j utilizzano un driver di routing e specificano una modalità di accesso per instradare le richieste di lettura e scrittura a un server appropriato in un cluster causale.

Al momento di eseguire la migrazione di un'applicazione client su Neptune, usa gli endpoint Neptune per instradare le query in modo efficace verso un'istanza appropriata nel tuo cluster:

- Tutte le connessioni a Neptune devono utilizzare `bolt://` anziché `bolt+routing://` o `neo4j://` nell'URL.
- L'endpoint del cluster consente la connessione all'istanza primaria corrente nel cluster. Usa l'endpoint del cluster per instradare le richieste di scrittura all'istanza primaria.
- L'endpoint del reader distribuisce le connessioni tra le istanze di replica di lettura nel cluster. Se hai un cluster a istanza singola senza istanze di replica di lettura, l'endpoint del reader si connette all'istanza primaria, che supporta le operazioni di scrittura. Se il cluster contiene una o più istanze di replica di lettura, l'invio di una richiesta di scrittura all'endpoint di lettura genera un'eccezione.
- Ogni istanza del cluster può anche avere il proprio endpoint. Utilizza un endpoint di istanza se l'applicazione client deve inviare una richiesta a un'istanza specifica del cluster.

Per ulteriori informazioni, consulta Considerazioni sugli endpoint Neptune.

Coerenza dei dati in Neptune

Se si utilizzano cluster causali Neo4j, le repliche di lettura saranno coerenti con i server principali, ma le applicazioni client possono garantire la coerenza causale utilizzando il concatenamento causale. Questo tipo di concatenamento prevede lo scambio di segnalibri tra le transazioni, il che consente a un'applicazione client di scrivere su un server principale e quindi leggere ciò che ha scritto in una replica di lettura.

In Neptune, le istanze di replica di lettura saranno coerenti con l'istanza writer. Il ritardo di replica di solito è inferiore a 100 millisecondi. Tuttavia, finché non viene replicata una modifica, gli aggiornamenti agli archi e ai vertici esistenti, nonché le aggiunte di nuovi archi e vertici, non sono visibili su un'istanza di replica. Pertanto, se l'applicazione richiede una coerenza immediata su Neptune mediante la lettura di ogni scrittura, usa l'endpoint del cluster per l'operazione di lettura dopo la scrittura. Questo è l'unico caso in cui è possibile utilizzare l'endpoint del cluster per operazioni di lettura. In tutte le altre circostanze, utilizza l'endpoint del reader per le letture.

Migrazione delle query da Neo4j a Neptune

Sebbene il supporto di openCypher in Neptune riduca drasticamente la quantità di lavoro richiesta per eseguire la migrazione delle query da Neo4j, esistono alcune differenze da valutare durante la migrazione:

- Come detto in precedenza in Ottimizzazioni del modello di dati, potrebbe essere necessario apportare modifiche al modello di dati in modo da creare un modello di dati a grafo ottimizzato per Neptune, il che prevede altre modifiche alle query e ai test.
- Neo4j offre una serie di estensioni di linguaggio specifiche per Cypher, non incluse nelle specifiche openCypher implementate da Neptune. A seconda del caso d'uso e della funzionalità utilizzata, potrebbero esistere soluzioni alternative all'interno del linguaggio openCypher oppure mediante l'utilizzo del linguaggio Gremlin o di altri meccanismi come descritto in Riscrittura delle query Cypher da eseguire in openCypher su Neptune.
- Invece dei driver Bolt, le applicazioni utilizzano spesso altri componenti middleware per interagire con il database. Consulta Compatibilità di Neptune con Neo4j per verificare se gli strumenti o il middleware che stai utilizzando sono supportati.
- In caso di failover, il driver Bolt potrebbe continuare a connettersi all'istanza writer o reader precedente perché l'endpoint del cluster fornito alla connessione è stato risolto in un indirizzo IP. Una corretta gestione degli errori nell'applicazione dovrebbe risolvere questo problema, come descritto in Creazione di una nuova connessione dopo il failover.
- Quando le transazioni vengono annullate a causa di conflitti irrisolvibili o timeout di attesa del blocco, Neptune risponde con un'eccezione `ConcurrentModificationException`. Per ulteriori informazioni, consulta Codici di errore del motore. Come best practice, i client devono sempre acquisire e gestire queste eccezioni.

  A volte si verifica un'eccezione `ConcurrentModificationException` quando più thread o più applicazioni scrivono contemporaneamente sul sistema. A causa dei livelli di isolamento delle transazioni, in alcuni casi questi conflitti possono essere inevitabili.
- Neptune supporta l'esecuzione di query Gremlin e openCypher sugli stessi dati. Ciò significa che in alcuni scenari potrebbe essere necessario prendere in considerazione l'utilizzo di Gremlin, che offre funzionalità di query più avanzate, per eseguire alcune delle funzionalità delle tue query.

Come detto in precedenza in Provisioning dell'infrastruttura, ogni applicazione deve essere sottoposta ad attività per trovare le dimensioni corrette al fine di garantire che il numero di istanze,

le loro dimensioni e la topologia del cluster siano tutti ottimizzati per il carico di lavoro specifico dell'applicazione.

Qui vengono trattati gli aspetti più comuni della migrazione dell'applicazione, ma ne esistono alche altri. Ogni applicazione è univoca. Contatta l'assistenza AWS o il team del tuo account se hai altre domande.

Migrazione di funzionalità e strumenti specifici per Neo4j

Neo4j ha una serie di funzionalità personalizzate e componenti aggiuntivi con funzionalità su cui l'applicazione potrebbe basarsi. Quando valuti la necessità di eseguire la migrazione di queste funzionalità, spesso è utile verificare se AWS offre un approccio migliore per raggiungere lo stesso obiettivo. Considerando le differenze architettoniche tra Neo4j e Neptune, spesso è possibile trovare alternative efficaci che sfruttano altri servizi AWS o integrazioni.

Consulta Compatibilità di Neptune con Neo4j per un l'elenco delle funzionalità specifiche di Neo4J e delle soluzioni alternative suggerite.

Compatibilità di Neptune con Neo4j

Neo4j si basa su un approccio all-in-one in cui il caricamento dei dati, l'ETL dei dati, le query delle applicazioni, l'archiviazione di dati e le operazioni di gestione avvengono tutti nello stesso set di risorse di calcolo, ad esempio le istanze EC2. Amazon Neptune è un database a grafo con specifiche aperte, incentrato su OLTP e in cui l'architettura separa le operazioni e disaccoppia le risorse in modo che possano essere dimensionate in modo dinamico.

Esistono diverse funzionalità e strumenti in Neo4j, inclusi strumenti di terze parti, che non fanno parte delle specifiche openCypher, non sono compatibili con openCypher o non sono compatibili con l'implementazione di openCypher di Neptune. Di seguito sono elencati alcuni dei più comuni.

Funzionalità specifiche di Neo4J non presenti in Neptune

- **`LOAD CSV`**: Neptune ha un approccio architettonico diverso per quel che riguarda il caricamento dei dati rispetto a Neo4j. Per consentire un migliore dimensionamento e l'ottimizzazione dei costi, Neptune implementa una separazione dei problemi relativi alle risorse e consiglia di utilizzare una delle integrazioni dei servizi AWS, ad esempio AWS Glue, per eseguire i processi ETL richiesti per preparare i dati in un formato supportato dallo strumento di caricamento in blocco Neptune.

  Per ottenere gli stessi risultati, un'altra opzione è utilizzare il codice dell'applicazione in esecuzione su risorse di calcolo AWS come istanze Amazon EC2, funzioni Lambda, Amazon Elastic Container Service, processi AWS Batch e così via. Il codice può utilizzare l'endpoint HTTPS di Neptune o l'endpoint Bolt.
- Controllo granulare degli accessi: Neptune supporta il controllo granulare degli accessi sulle azioni di accesso ai dati utilizzando le chiavi di condizione IAM. È possibile implementare un ulteriore controllo granulare degli accessi a livello di applicazione.
- Neo4j Fabric: Neptune supporta la federazione delle query tra database per carichi di lavoro RDF utilizzando la parola chiave SPARQL `SERVICE`. Poiché attualmente non esiste uno standard o una specifica aperta per la federazione delle query per carichi di lavoro a grafo delle proprietà, questa funzionalità dovrebbe essere implementata a livello di applicazione.
- Controllo degli accessi basato sui ruoli (RBAC): Neptune gestisce l'autenticazione attraverso l'assegnazione di policy e ruoli IAM. Le policy e i ruoli IAM forniscono un livello estremamente flessibile di gestione degli utenti all'interno dell'applicazione, quindi è importante leggere e comprendere le informazioni nella panoramica di IAM prima di configurare il cluster.
- Aggiunta di segnalibri: i cluster Neptune sono costituiti da una singola istanza writer e da un massimo di 15 istanze di replica di lettura. I dati scritti sull'istanza writer sono conformi agli

standard ACID e garantiscono in modo efficace la coerenza delle letture successive. Le repliche di lettura utilizzano lo stesso volume di archiviazione dell'istanza writer e saranno coerenti, in genere entro meno di 100 ms dal momento in cui vengono scritti i dati. Se il tuo caso d'uso prevede un'esigenza immediata di garantire la coerenza per la lettura delle nuove scritture, queste ultime devono essere indirizzate all'endpoint del cluster anziché all'endpoint del reader.

- Procedure APOC: poiché le procedure APOC non sono incluse nelle specifiche openCypher, Neptune non fornisce supporto diretto per procedure esterne. Neptune si affida invece alle integrazioni con altri servizi AWS per ottenere funzionalità simili per l'utente finale in modo scalabile, sicuro e affidabile. A volte le procedure APOC possono essere riscritte in openCypher o Gremlin e alcune non sono pertinenti per le applicazioni Neptune.

  In generale, le procedure APOC rientrano nelle seguenti categorie:

  - Importazione: Neptune supporta l'importazione dei dati mediante una serie di formati utilizzando linguaggi di query, lo strumento di caricamento in blocco Neptune o come destinazione di AWS Database Migration Service. Le operazioni ETL sui dati potrebbero essere eseguite utilizzando AWS Glue e il pacchetto open source `neptune-python-utils`.
  - Esportazione: Neptune supporta l'esportazione dei dati utilizzando l'utilità `neptune-export`, che supporta una serie di formati e metodi di esportazione comuni.
  - Integrazione dei database: Neptune supporta l'integrazione con altri database utilizzando strumenti ETL come AWS Glue o strumenti di migrazione come AWS Database Migration Service.
  - Aggiornamenti dei grafi: Neptune supporta un ampio set di funzionalità per l'aggiornamento dei dati a grafo delle proprietà attraverso il supporto per i linguaggi di query openCypher e Gremlin. Consulta Riscritture Cypher per alcuni esempi di riscritture di procedure di uso comune.
  - Strutture dei dati: Neptune supporta un ampio set di funzionalità per l'aggiornamento dei dati a grafo delle proprietà attraverso il supporto per i linguaggi di query openCypher e Gremlin. Consulta Riscritture Cypher per alcuni esempi di riscritture di procedure di uso comune.
  - Procedure temporali (data e ora): Neptune supporta un ampio set di funzionalità per l'aggiornamento dei dati a grafo delle proprietà attraverso il supporto per i linguaggi di query openCypher e Gremlin. Consulta Riscritture Cypher per alcuni esempi di riscritture di procedure di uso comune.
  - Procedure matematiche: Neptune supporta un ampio set di funzionalità per l'aggiornamento dei dati a grafo delle proprietà attraverso il supporto per i linguaggi di query openCypher e Gremlin. Consulta Riscritture Cypher per alcuni esempi di riscritture di procedure di uso comune.

  - Query avanzate sui grafi: Neptune supporta un ampio set di funzionalità per l'aggiornamento dei dati a grafo delle proprietà attraverso il supporto per i linguaggi di query openCypher e Gremlin. Consulta Riscritture Cypher per alcuni esempi di riscritture di procedure di uso comune.
  - Confronto dei grafi: Neptune supporta un ampio set di funzionalità per l'aggiornamento dei dati a grafo delle proprietà attraverso il supporto per i linguaggi di query openCypher e Gremlin. Consulta Riscritture Cypher per alcuni esempi di riscritture di procedure di uso comune.
  - Esecuzione di Cypher: Neptune supporta un ampio set di funzionalità per l'aggiornamento dei dati a grafo delle proprietà attraverso il supporto per i linguaggi di query openCypher e Gremlin. Consulta Riscritture Cypher per alcuni esempi di riscritture di procedure di uso comune.
- Procedure personalizzate: Neptune non supporta procedure personalizzate create dagli utenti. Questa funzionalità dovrebbe essere implementata a livello di applicazione.
- Procedure geospaziali: sebbene Neptune non fornisca un supporto nativo per le funzionalità geospaziali, è possibile ottenere funzionalità simili attraverso l'integrazione con altri servizi AWS, come spiegato nel post del blog Combine Amazon Neptune and Amazon OpenSearch Service for geospatial queries di Ross Gabay e Abhilash Vinod (1 febbraio 2022).
- Data Science grafici: Neptune supporta l'analisi di grafici tramite Analisi Neptune, un motore ottimizzato per la memoria che supporta una libreria di algoritmi di analisi di grafici.

  Neptune fornisce anche un'integrazione con l'SDK AWS Pandas e diversi notebook di esempio che mostrano come sfruttare questa integrazione all'interno degli ambienti Python per eseguire analisi su dati a grafo.

- Vincoli di schema: all'interno di Neptune, l'unico vincolo di schema disponibile è l'unicità dell'ID di un nodo o un arco. Non esiste alcuna funzionalità che consenta di specificare altri vincoli di schema o altri vincoli di unicità o valore su un elemento nel grafo. I valori degli ID in Neptune sono stringhe e possono essere impostati usando Gremlin, come descritto di seguito:

  ```
  g.addV('person').property(id, '1') )
  ```

  Le applicazioni che devono utilizzare l'ID come vincolo di unicità dovrebbero prendere in considerazione questo approccio per ottenere un vincolo di unicità. Se l'applicazione utilizza più colonne come vincolo di unicità, l'ID può essere impostato su una combinazione di questi valori. Ad esempio, `id=123, code='SEA'` potrebbe essere rappresentato come `ID='123_SEA'` per ottenere un vincolo di unicità complesso.
- Multi-tenancy: Neptune supporta un solo grafo per cluster. Per creare un sistema multi-tenant mediante Neptune, utilizza più cluster o partiziona logicamente i tenant all'interno di un singolo

grafo e utilizza la logica dell'applicazione per implementare la separazione. Ad esempio, aggiungi una proprietà `tenantId` e includila in ogni query, in questo modo:

```
MATCH p=(n {tenantId:1})-[]->({tenantId:1}) RETURN p LIMIT 5)
```

Neptune Serverless semplifica l'implementazione multi-tenancy utilizzando più cluster database, ognuno dei quali viene dimensionato in modo indipendente e automatico in base alle esigenze.

Supporto di Neptune per gli strumenti Neo4j

Neptune fornisce le seguenti alternative agli strumenti Neo4j:

- Neo4j Browser: Neptune fornisce notebook per grafi open source che forniscono un IDE mirato agli sviluppatori per eseguire le query e visualizzare i risultati.
- Neo4j Bloom: Neptune supporta visualizzazioni avanzate dei grafi utilizzando soluzioni di visualizzazione di terze parti come Graph-explorer, Tom Sawyer, Cambridge Intelligence, Graphistry, metaphacts e G.V().
- GraphQL: Neptune al momento supporta GraphQL tramite integrazioni AWS AppSync personalizzate. Consulta il post del blog Build a graph application with Amazon Neptune and AWS Amplify e il progetto di esempio Building Serverless Calorie tracker application with AWS AppSync and Amazon Neptune.
- NeoSemantics: Neptune supporta nativamente il modello di dati RDF, quindi ai clienti che desiderano eseguire carichi di lavoro RDF si consiglia di utilizzare il supporto del modello RDF di Neptune.
- Arrows.app: il Cypher creato durante l'esportazione del modello utilizzando il comando di esportazione è compatibile con Neptune.
- Linkurious Ogma: qui è disponibile un esempio di integrazione con Linkurious Ogma.
- Spring Data Neo4j: al momento non è compatibile con Neptune.
- Connettore Spark di Neo4j: il connettore Spark di Neo4j può essere utilizzato all'interno di un processo Spark per connettersi a Neptune utilizzando openCypher. Ecco alcuni esempi di codice e configurazione dell'applicazione:

  Esempio di codice:

```
SparkSession spark = SparkSession
            .builder()
```

```
            .config("encryption.enabled", "true")
            .appName("Simple Application").config("spark.master",
 "local").getOrCreate();

Dataset<Row> df = spark.read().format("org.neo4j.spark.DataSource")
            .option("url", "bolt://(your cluster endpoint):8182")
            .option("encryption.enabled", "true")
            .option("query", "MATCH (n:airport) RETURN n")
            .load();

System.out.println("TOTAL RECORD COUNT: " + df.count());
spark.stop();
```

Configurazione dell'applicazione:

```
<dependency>
    <groupId>org.neo4j</groupId>
    <artifactId>neo4j-connector-apache-spark_2.12-4.1.0</artifactId>
    <version>4.0.1_for_spark_3</version>
</dependency>
```

Funzionalità e strumenti di Neo4j non elencati qui

Se utilizzi uno strumento o una funzionalità non elencati qui, non ne garantiamo la compatibilità con Neptune o altri servizi all'interno di AWS. Contatta l'assistenza AWS o il team del tuo account se hai altre domande.

Riscrittura delle query Cypher da eseguire in openCypher su Neptune

openCypher è un linguaggio di query dichiarativo per grafi di proprietà che è stato sviluppato originariamente da Neo4j, per poi diventare open source nel 2015, e che ha contribuito al progetto openCypher con una licenza open source Apache 2. In AWS, crediamo che l'open source sia un bene per tutti e ci impegniamo a offrirne il valore ai nostri clienti, nonché a fornire l'eccellenza operativa di AWS alle community open source.

La sintassi di openCypher è documentata nel documento Cypher Query Language Reference, versione 9.

Poiché openCypher contiene un sottoinsieme della sintassi e delle caratteristiche del linguaggio di query Cypher, alcuni scenari di migrazione richiedono la riscrittura delle query in moduli compatibili con openCypher oppure la valutazione di metodi alternativi per ottenere la funzionalità desiderata.

Questa sezione contiene alcuni consigli non esaustivi per gestire le differenze più comuni. È necessario testare accuratamente qualsiasi applicazione utilizzando queste riscritture per assicurarsi che i risultati siano quelli previsti.

Riscrittura delle funzioni di predicato **`None`**, **`All`** e **`Any`**

Queste funzioni non fanno parte della specifica openCypher. È possibile ottenere risultati simili in openCypher utilizzando la comprensione di lista.

Ad esempio, trova tutti i percorsi che vanno dal nodo `Start` al nodo `End`, ma nessun percorso può attraversare un nodo con una proprietà di classe `D`:

```
# Neo4J Cypher code
match p=(a:Start)-[:HOP*1..]->(z:End)
where none(node IN nodes(p) where node.class ='D')
return p

# Neptune openCypher code
match p=(a:Start)-[:HOP*1..]->(z:End)
where size([node IN nodes(p) where node.class = 'D']) = 0
return p
```

La comprensione di lista consente di ottenere questi risultati come descritto di seguito:

```
all  => size(list_comprehension(list)) = size(list)
```

```
any  => size(list_comprehension(list)) >= 1
none => size(list_comprehension(list)) = 0
```

Riscrittura della funzione Cypher **`reduce()`** in openCypher

La funzione `reduce()` non fa parte delle specifiche openCypher. Viene spesso utilizzata per creare un'aggregazione di dati a partire da elementi all'interno di un elenco. In molti casi, è possibile utilizzare una combinazione di comprensione di lista e della clausola `UNWIND` per ottenere risultati simili.

Ad esempio, la seguente query Cypher trova tutti gli aeroporti su rotte che comprendono da uno a tre scali Anchorage (ANC) e Austin (AUS) e restituisce la distanza totale di ciascun percorso:

```
MATCH p=(a:airport {code: 'ANC'})-[r:route*1..3]->(z:airport {code: 'AUS'})
RETURN p, reduce(totalDist=0, r in relationships(p) | totalDist + r.dist) AS totalDist
ORDER BY totalDist LIMIT 5
```

Puoi scrivere la stessa query in openCypher per Neptune come segue:

```
MATCH p=(a:airport {code: 'ANC'})-[r:route*1..3]->(z:airport {code: 'AUS'})
UNWIND [i in relationships(p) | i.dist] AS di
RETURN p, sum(di) AS totalDist
ORDER BY totalDist
LIMIT 5
```

Riscrittura della clausola Cypher FOREACH in openCypher

La clausola FOREACH non fa parte delle specifiche openCypher. Viene spesso utilizzata per aggiornare i dati in mezzo a una query, spesso da aggregazioni o elementi all'interno di un percorso.

Come esempio di percorso, trova tutti gli aeroporti su una rotta con non più di due scali tra Anchorage (ANC) e Austin (AUS) e imposta la proprietà Visited per ciascuno di essi:

```
# Neo4J Example
MATCH p=(:airport {code: 'ANC'})-[*1..2]->({code: 'AUS'})
FOREACH (n IN nodes(p) | SET n.visited = true)

# Neptune openCypher
MATCH p=(:airport {code: 'ANC'})-[*1..2]->({code: 'AUS'})
WITH nodes(p) as airports
UNWIND airports as a
```

```
SET a.visited=true
```

Un altro esempio è:

```
# Neo4J Example
MATCH p=(start)-[*]->(finish)
WHERE start.name = 'A' AND finish.name = 'D'
FOREACH (n IN nodes(p) | SET n.marked = true)

# Neptune openCypher
MATCH p=(start)-[*]->(finish)
WHERE start.name = 'A' AND finish.name = 'D'
UNWIND nodes(p) AS n
SET n.marked = true
```

Riscrittura delle procedure APOC Neo4j in Neptune

Gli esempi seguenti utilizzano openCypher per sostituire alcune delle procedure APOC usate più comunemente. Questi esempi sono solo di riferimento e hanno lo scopo di fornire alcuni suggerimenti su come gestire scenari comuni. In pratica, ogni applicazione è diversa ed è necessario trovare strategie personalizzate per fornire tutte le funzionalità di cui hai bisogno.

Riscrittura delle procedure **`apoc.export`**

Neptune offre un array di opzioni sia per le esportazioni a grafo completo che per quelle basate su query in vari formati di output come CSV e JSON, il tutto mediante l'utilità neptune-export (vedi Esportazione di dati da un cluster database Neptune).

Riscrittura delle procedure **`apoc.schema`**

Neptune non ha schemi, indici o vincoli definiti in modo esplicito, perciò molte procedure `apoc.schema` non sono più necessarie. Alcuni esempi sono:

- `apoc.schema.assert`
- `apoc.schema.node.constraintExists`
- `apoc.schema.node.indexExists`,
- `apoc.schema.relationship.constraintExists`
- `apoc.schema.relationship.indexExists`
- `apoc.schema.nodes`

- `apoc.schema.relationships`

openCypher per Neptune supporta il recupero di valori simili a quelli utilizzati dalle procedure, come mostrato di seguito, ma potrebbe riscontrare problemi di prestazioni su grafi più grandi poiché sarebbe necessario scansionare un'ampia porzione del grafo per restituire la risposta.

```
# openCypher replacement for apoc.schema.properties.distinct
MATCH (n:airport)
RETURN DISTINCT n.runways
```

```
# openCypher replacement for apoc.schema.properties.distinctCount
MATCH (n:airport)
RETURN DISTINCT n.runways, count(n.runways)
```

Alternative alle procedure **`apoc.do`**

Queste procedure vengono utilizzate per consentire l'esecuzione di query condizionali facili da implementare utilizzando altre clausole openCypher. In Neptune ci sono almeno due modi per ottenere un comportamento simile:

- Un modo è combinare le funzionalità della comprensione di lista di openCypher con la clausola `UNWIND`.
- Un altro modo è usare i passaggi choose() e coalesce() in Gremlin.

Di seguito sono riportati alcuni esempi di questi approcci.

Alternative ad apoc.do.when

```
# Neo4J Example
MATCH (n:airport {region: 'US-AK'})
CALL apoc.do.when(
 n.runways>=3,
 'SET n.is_large_airport=true RETURN n',
 'SET n.is_large_airport=false RETURN n',
 {n:n}
) YIELD value
WITH collect(value.n) as airports
RETURN size([a in airports where a.is_large_airport]) as large_airport_count,
size([a in airports where NOT a.is_large_airport]) as small_airport_count
```

```
# Neptune openCypher
MATCH (n:airport {region: 'US-AK'})
WITH n.region as region, collect(n) as airports
WITH [a IN airports where a.runways >= 3] as large_airports,
[a IN airports where a.runways < 3] as small_airports, airports
UNWIND large_airports as la
SET la.is_large_airport=true
WITH DISTINCT small_airports, airports
UNWIND small_airports as la
    SET la.small_airports=true
WITH DISTINCT airports
RETURN size([a in airports where a.is_large_airport]) as large_airport_count,
size([a in airports where NOT a.is_large_airport]) as small_airport_count

#Neptune Gremlin using choose()
g.V().
  has('airport', 'region', 'US-AK').
  choose(
    values('runways').is(lt(3)),
    property(single, 'is_large_airport', false),
    property(single, 'is_large_airport', true)).
  fold().
  project('large_airport_count', 'small_airport_count').
    by(unfold().has('is_large_airport', true).count()).
    by(unfold().has('is_large_airport', false).count())

 #Neptune Gremlin using coalesce()
g.V().
  has('airport', 'region', 'US-AK').
  coalesce(
    where(values('runways').is(lt(3))).
    property(single, 'is_large_airport', false),
    property(single, 'is_large_airport', true)).
  fold().
  project('large_airport_count', 'small_airport_count').
    by(unfold().has('is_large_airport', true).count()).
    by(unfold().has('is_large_airport', false).count())
```

Alternative ad apoc.do.case

```
# Neo4J Example
```

```
MATCH (n:airport {region: 'US-AK'})
CALL apoc.case([
 n.runways=1, 'RETURN "Has one runway" as b',
 n.runways=2, 'RETURN "Has two runways" as b'
 ],
 'RETURN "Has more than 2 runways" as b'
) YIELD value
RETURN {type: value.b,airport: n}

# Neptune openCypher
MATCH (n:airport {region: 'US-AK'})
WITH n.region as region, collect(n) as airports
WITH [a IN airports where a.runways =1] as single_runway,
[a IN airports where a.runways =2] as double_runway,
[a IN airports where a.runways >2] as many_runway
UNWIND single_runway as sr
    WITH {type: "Has one runway",airport: sr} as res, double_runway, many_runway
WITH DISTINCT double_runway as double_runway, collect(res) as res, many_runway
UNWIND double_runway as dr
    WITH {type: "Has two runways",airport: dr} as two_runways, res, many_runway
WITH collect(two_runways)+res as res, many_runway
UNWIND many_runway as mr
    WITH {type: "Has more than 2 runways",airport: mr} as res2, res, many_runway
WITH collect(res2)+res as res
UNWIND res as r
RETURN r

#Neptune Gremlin using choose()
g.V().
  has('airport', 'region', 'US-AK').
  project('type', 'airport').
    by(
      choose(values('runways')).
        option(1, constant("Has one runway")).
        option(2, constant("Has two runways")).
        option(none, constant("Has more than 2 runways"))).
    by(elementMap())

 #Neptune Gremlin using coalesce()
 g.V().
  has('airport', 'region', 'US-AK').
  project('type', 'airport').
    by(
      coalesce(
```

```
        has('runways', 1).constant("Has one runway"),
        has('runways', 2).constant("Has two runways"),
        constant("Has more than 2 runways"))).
    by(elementMap())
```

Alternative alle proprietà basate su elenco

Neptune attualmente non supporta la memorizzazione di proprietà basate su elenco. Tuttavia, è possibile ottenere risultati simili memorizzando i valori dell'elenco come stringa separata da virgole e, successivamente, utilizzando le funzioni `join()` e `split()` per costruire e decostruire la proprietà di elenco.

Ad esempio, per salvare un elenco di tag come proprietà, potremmo usare l'esempio di riscrittura che mostra come recuperare una proprietà separata da virgole, quindi utilizzare le funzioni `split()` e `join()` con la comprensione di lista per ottenere risultati simili:

```
# Neo4j Example (In this example, tags is a durable list of string.
MATCH (person:person {name: "TeeMan"})
WITH person, [tag in person.tags WHERE NOT (tag IN ['test1', 'test2', 'test3'])] AS
 newTags
SET person.tags = newTags
RETURN person

# Neptune openCypher
MATCH (person:person {name: "TeeMan"})
WITH person, [tag in split(person.tags, ',') WHERE NOT (tag IN ['test1', 'test2',
 'test3'])] AS newTags
SET person.tags = join(newTags,',')
RETURN person
```

Risorse per la migrazione da Neo4j a Neptune

Neptune fornisce diversi strumenti e risorse utili nel processo di migrazione.

Strumenti utili per la migrazione da Neo4j a Neptune

- Il CheatSheet di openCypher.
- neo4j-to-neptune: un'utilità a riga di comando per la migrazione dei dati da Neo4j a Neptune.
- fully-automated-neo4j-to-neptune: un'applicazione AWS CDK che mostra come eseguire la migrazione di semplici database Neo4j su Amazon Neptune.
- csv-to-neptune-bulk-format: questo strumento adotta un approccio basato sulla configurazione per riformattare uno o più file CSV in un formato di caricamento in blocco supportato da Neptune.

Post del blog

- Change data capture from Neo4j to Amazon Neptune using Amazon Managed Streaming for Apache Kafka di Sanjeet Sahay (22 giugno 2020)
- Migrating a Neo4j graph database to Amazon Neptune with a fully automated utility di Sanjeet Sahay (13 aprile 2020)

Migrazione di un grafo esistente da un server Apache TinkerPop Gremlin ad Amazon Neptune

Ecco come procedere per eseguire la migrazione di dati dei grafi esistenti da un server Apache TinkerPop Gremlin ad Amazon Neptune:

1. Esporta i dati dal server Gremlin ad Amazon Simple Storage Service (Amazon S3).
2. Converti i dati esportati in un formato CSV supportato dallo strumento di caricamento in blocco Neptune.
3. Utilizza Strumento di caricamento in blocco Neptune per importare i dati in un cluster database Neptune che hai già preparato.
4. Modifica l'applicazione esistente in modo che si connetta all'endpoint Gremlin di Neptune e apporta le modifiche necessarie per far fronte alle differenze di implementazione tra Neptune e Gremlin.

Migrazione di un grafo esistente da un triplestore RDF ad Amazon Neptune

Ecco come procedere per eseguire la migrazione di dati dei grafi esistenti da un triplestore RDF/SPARQL ad Amazon Neptune:

1. Esporta i dati dal triplestore RDF.
2. Converti i dati esportati in un formato supportato dallo strumento di caricamento in blocco Neptune.
3. Archivia i dati da importare in Amazon Simple Storage Service (Amazon S3).
4. Utilizza Strumento di caricamento in blocco Neptune per importare i dati di Amazon S3 in un cluster database Neptune che hai già preparato.
5. Modifica l'applicazione esistente in modo che si connetta all'endpoint SPARQL di Neptune.

Se vuoi provare a eseguire la migrazione dei dati CSV dei grafi di proprietà in RDF, puoi utilizzare il convertitore da CSV a RDF di Amazon Neptune.

Utilizzo di AWS Database Migration Service (AWS DMS) per eseguire la migrazione da un database relazionale o NoSQL ad Amazon Neptune

AWS Database Migration Service (AWS DMS) è un servizio cloud che consente di eseguire la migrazione di database relazionali, data warehouse, database NoSQL e altri tipi di datastore. Se disponi di dati dei grafi archiviati in uno dei database relazionali o NoSQL supportati da AWS DMS, AWS DMS può aiutarti a eseguire la migrazione a Neptune in modo rapido e sicuro, senza che il database attuale debba essere soggetto a tempi di inattività. Per informazioni dettagliate, consulta Utilizzo AWS Database Migration Service per caricare dati in Amazon Neptune da un altro data store.

Il flusso per la migrazione mediante AWS DMS è il seguente:

- Crea un oggetto di mappatura delle tabelle AWS DMS. Questo oggetto JSON specifica quali tabelle devono essere lette dal database di origine, l'ordine con cui vengono lette e come vengono denominate le colonne. Può anche filtrare le righe copiate e fornire semplici trasformazioni di valore come la conversione in minuscolo o l'arrotondamento.
- Crea un oggetto `GraphMappingConfig` di Neptune per specificare in che modo i dati estratti dal database di origine devono essere caricati in Neptune.
  - Per i dati RDF (interrogati utilizzando SPARQL), `GraphMappingConfig` viene scritto nel linguaggio di mappatura R2RML standard di W3.
  - Per i dati dei grafi di proprietà (sottoposti a query mediante Gremlin), `GraphMappingConfig` è un oggetto JSON, come descritto in GraphMappingConfig Layout per dati Property-Graph/Gremlin.
- Crea un'istanza di replica AWS DMS nello stesso VPC del cluster database Neptune per eseguire la migrazione.
- Crea un bucket Amazon S3 da utilizzare come archiviazione intermedia per lo staging dei dati di cui eseguire la migrazione.
- Esegui l'attività di migrazione di AWS DMS.

Per i dettagli, consulta Utilizzo AWS Database Migration Service per caricare dati in Amazon Neptune da un altro data store e anche il post del blog (in quattro parti) di Chris Smith da titolo "Populating your graph in Amazon Neptune from a relational database using AWS Database Migration Service (DMS):"

- Part 1: Setting the stage
- Part 2: Designing the property graph model
- Part 3: Designing the RDF Model
- Part 4: Putting it all together

Migrazione da Blazegraph ad Amazon Neptune

Se esiste un grafo nel triplestore RDF open source Blazegraph, puoi eseguire la migrazione dei dati dei grafi su Amazon Neptune seguendo questa procedura:

- Esegui il provisioning dell'infrastruttura AWS. Per iniziare, esegui il provisioning dell'infrastruttura Neptune richiesta utilizzando un modello CloudFormation AWS (vedi Creazione di un cluster DB).
- Esporta i dati da Blazegraph. Esistono due metodi principali per esportare i dati da Blazegraph: utilizzando le query CONSTRUCT di SPARQL o mediante l'utilità Blazegraph Export.
- Importa i dati in Neptune. Successivamente, puoi caricare i file di dati esportati in Neptune utilizzando il workbench di Neptune e lo Strumento di caricamento in blocco Neptune.

Questo approccio di solito è applicabile anche per la migrazione da altri database triplestore RDF.

Compatibilità tra Blazegraph e Neptune

Prima di eseguire la migrazione dei dati dei grafi su Neptune, è bene conoscere alcune differenze significative tra Blazegraph e Neptune. Questi aspetti diversi possono richiedere modifiche alle query, all'architettura dell'applicazione oppure a entrambi o addirittura rendere la migrazione poco pratica:

- **`Full-text search`**: in Blazegraph, puoi utilizzare funzionalità di ricerca full-text interne o esterne tramite un'integrazione con Apache Solr. Se utilizzi una di queste funzionalità, resta al passo con gli ultimi aggiornamenti sulla ricerca full-text supportata da Neptune. Per informazioni, consultare Ricerca full-text di Neptune.
- **`Query hints`**: sia Blazegraph che Neptune estendono SPARQL mediante il concetto dei suggerimenti delle query. Durante una migrazione, è necessario eseguire trasferire tutti i suggerimenti delle query che utilizzi. Per informazioni sugli ultimi suggerimenti delle query supportati da Neptune, consulta Hint di query SPARQL.
- Inferenza: Blazegraph supporta l'inferenza come opzione configurabile in modalità tripla, ma non in modalità quadrupla. Neptune non supporta ancora l'inferenza.
- Ricerca geospaziale: Blazegraph supporta la configurazione di spazi dei nomi che abilitano il supporto geospaziale. Questa funzionalità non è ancora disponibile in Neptune.
- Multi-tenancy: Blazegraph supporta la multi-tenancy all'interno di un singolo database. In Neptune, la multi-tenancy è supportata sia mediante l'archiviazione dei dati in grafi denominati sia utilizzando le clausole USING NAMED per le query SPARQL oppure creando un cluster database separato per ogni tenant.

- Federazione: Neptune attualmente supporta la federazione SPARQL 1.1 in posizioni a cui può accedere l'istanza Neptune, ad esempio all'interno del VPC privato, tra più VPC o in endpoint Internet esterni. A seconda della configurazione specifica e degli endpoint della federazione richiesti, potrebbe essere necessaria una configurazione di rete aggiuntiva.
- Estensioni degli standard Blazegraph: Blazegraph include più estensioni per gli standard SPARQL e REST API, mentre Neptune è compatibile solo con le specifiche di questi standard. Ciò potrebbe richiedere modifiche all'applicazione o rendere difficile la migrazione.

Provisioning dell'infrastruttura AWS per Neptune

Sebbene sia possibile creare manualmente l'infrastruttura AWS richiesta tramite la AWS Management Console oAWS CLI, spesso è più comodo utilizzare un modello CloudFormation, come descritto di seguito:

Provisioning di Neptune mediante un modello CloudFormation:

1. Accedi a Utilizzo di uno AWS CloudFormation stack per creare un cluster Neptune DB.
2. Scegli Avvia lo stack nella tua regione preferita.
3. Imposta i parametri richiesti (nome dello stack e `EC2SSHKeyPairName`). Imposta anche i seguenti parametri facoltativi per facilitare il processo di migrazione:
   - Imposta `AttachBulkloadIAMRoleToNeptuneCluster` su true. Questo parametro consente di creare e associare il ruolo IAM appropriato al cluster per consentire il caricamento in blocco dei dati.
   - Imposta `NotebookInstanceType` il tipo di istanza preferito. Questo parametro crea una cartella di lavoro di Neptune da utilizzare per eseguire il caricamento in blocco su Neptune e convalidare la migrazione.
4. Seleziona Successivo.
5. Imposta tutte le altre opzioni che desideri per lo stack.
6. Seleziona Successivo.
7. Controlla le opzioni e seleziona entrambe le caselle di controllo per confermare di sapere che AWS CloudFormation potrebbe richiedere funzionalità aggiuntive.
8. Seleziona Crea stack.

Il processo può richiedere alcuni minuti.

Esportazione dei dati da Blazegraph.

Il passaggio successivo prevede l'esportazione dei dati da Blazegraph in un formato compatibile con lo strumento di caricamento in blocco Neptune.

A seconda del modo in cui i dati sono archiviati in Blazegraph (in modalità tripla o quadrupla) e del numero di grafi denominati in uso, Blazegraph potrebbe richiedere di eseguire il processo di esportazione più volte e generare più file di dati:

- Se i dati sono archiviati in modalità tripla, è necessario eseguire un'esportazione per ogni grafo denominato.
- Se i dati sono archiviati in modalità quadrupla, puoi scegliere di esportare i dati in formato N-Quads o esportare ogni grafo denominato in un formato N-Triples.

Di seguito si presuppone un singolo spazio di nomi venga esportato come N-Quads, ma è possibile ripetere la procedura per altri spazi di nomi o formati di esportazione desiderati.

Se hai bisogno che Blazegraph sia online e disponibile durante la migrazione, usa le query CONSTRUCT di SPARQL. Ciò richiede l'installazione, la configurazione e l'esecuzione di un'istanza Blazegraph con un endpoint SPARQL accessibile.

Se non hai bisogno che Blazegraph sia online, usa l'utilità BlazeGraph Export. A questo scopo, oltre a scaricare Blazegraph, il file di dati e i file di configurazione devono essere accessibili, ma non è necessario che il server sia in esecuzione.

Esportazione dei dati da Blazegraph mediante CONSTRUCT di SPARQL

CONSTRUCT è una funzionalità di SPARQL che restituisce un grafo RDF corrispondente a un modello di query specificato. In questo caso d'uso, viene utilizzata per esportare i dati di uno spazio dei nomi alla volta, utilizzando una query simile a questa:

```
CONSTRUCT WHERE { hint:Query hint:analytic "true" . hint:Query
 hint:constructDistinctSPO "false" . ?s ?p ?o }
```

Sebbene esistano altri strumenti RDF per esportare i dati, il modo più semplice per eseguire questa query è utilizzare l'endpoint della REST API fornito da Blazegraph. Lo script seguente mostra come usare uno script Python (3.6+) per esportare i dati in formato N-Quads:

```
import requests
```

```
# Configure the URL here: e.g. http://localhost:9999/sparql
url = "http://localhost:9999/sparql"
payload = {'query': 'CONSTRUCT WHERE { hint:Query hint:analytic "true" . hint:Query
 hint:constructDistinctSPO "false" . ?s ?p ?o }'}
# Set the export format to be n-quads
headers = {
'Accept': 'text/x-nquads'
}
# Run the http request
response = requests.request("POST", url, headers=headers, data = payload, files = [])
#open the file in write mode, write the results, and close the file handler
f = open("export.nq", "w")
f.write(response.text)
f.close()
```

Se i dati sono archiviati in modalità tripla, devi modificare il parametro di intestazione `Accept` per esportare i dati in un formato appropriato (N-Triples, RDF/XML o Turtle) utilizzando i valori specificati nel repository GitHub di Blazegraph.

Esportazione dei dati mediante l'utilità di esportazione di Blazegraph

La classe `ExportKB` di Blazegraph è un'utilità per esportare i dati. `ExportKB` facilita l'esportazione dei dati da Blazegraph ma, a differenza del metodo precedente, richiede che il server sia offline durante l'esecuzione dell'esportazione. Per questo motivo è il metodo ideale se Blazegraph può rimanere offline durante la migrazione oppure se la migrazione può essere eseguita da un backup dei dati.

Esegui l'utilità da una riga di comando Java su una macchina in cui Blazegraph è installato, ma non è in esecuzione. Il modo più semplice per eseguire questo comando è scaricare l'ultima versione di blazegraph.jar che si trova su GitHub. L'esecuzione di questo comando richiede diversi parametri:

- **`log4j.primary.configuration`**: la posizione del file delle proprietà log4j.
- **`log4j.configuration`**: la posizione del file delle proprietà log4j.
- **`output`**: la directory di output per i dati esportati. I file si trovano in formato `tar.gz` in una sottodirectory denominata secondo quanto documentato nella knowledge base.
- **`format`**: il formato di output desiderato, seguito dalla posizione del file `RWStore.properties`. Se i dati sono in modalità tripla, devi cambiare il parametro `-format` impostandolo su `N-Triples`, `Turtle` o `RDF/XML`.

Ad esempio, nel caso del file del journal e dei file delle proprietà di Blazegraph, esporta i dati come N-Quads utilizzando il seguente codice:

```
java -cp blazegraph.jar \
    com.bigdata.rdf.sail.ExportKB \
    -outdir ~/temp/ \
    -format N-Quads \
    ./RWStore.properties
```

Se l'esportazione riesce, verrà visualizzato un output simile a questo:

```
Exporting kb as N-Quads on /home/ec2-user/temp/kb
Effective output directory: /home/ec2-user/temp/kb
Writing /home/ec2-user/temp/kb/kb.properties
Writing /home/ec2-user/temp/kb/data.nq.gz
Done
```

Creazione di un bucket Amazon Simple Storage Service (Amazon S3) e copia dei dati esportati al suo interno

Dopo aver esportato i dati da Blazegraph, crea un bucket Amazon Simple Storage Service (Amazon S3) nella stessa regione del cluster database Neptune di destinazione, in modo che lo strumento di caricamento in blocco Neptune possa usarlo come origine dell'importazione dei dati.

Per istruzioni su come creare un bucket Amazon S3, consulta l'argomento sulla creazione di un bucket S3 nella Guida per l'utente di Amazon Simple Storage Service, nonché gli esempi di creazione di un bucket nella Guida per l'utente di Amazon Simple Storage Service.

Per istruzioni su come copiare nel nuovo bucket Amazon S3 i file di dati che hai esportato, consulta l'argomento sul caricamento di un oggetto in un bucket nella Guida per l'utente di Amazon Simple Storage Service oppure scopri come usare i comandi di alto livello (s3) con AWS CLI. Per copiare i file uno alla volta, puoi anche usare un codice Python simile al seguente:

```
import boto3

region = 'region name'
bucket_name = 'bucket name'
s3 = boto3.resource('s3')
s3.meta.client.upload_file('export.nq', bucket_name, 'export.nq')
```

Uso dello strumento di caricamento in blocco di Neptune per importare i dati in Neptune

Dopo aver esportato i dati da Blazegraph e averli copiati in un bucket Amazon S3, è tutto pronto per importare i dati in Neptune. Lo strumento di caricamento in blocco di Neptune consente di caricare i dati più velocemente e con meno sovraccarico rispetto alle operazioni di caricamento con SPARQL. Il processo di caricamento in blocco viene avviato da una chiamata all'API dell'endpoint dello strumento di caricamento, in modo da caricare in Neptune i dati archiviati nel bucket S3 denominato.

Sebbene sia possibile farlo con una chiamata diretta all'endpoint REST dello strumento di caricamento, è necessario avere accesso al VPC privato in cui viene eseguita l'istanza di Neptune di destinazione. È possibile configurare un host bastione, accedere con SSH alla macchina ed eseguire il comando CURL, ma è più facile usare il workbench Neptune.

Il workbench Neptune è un notebook Jupyter preconfigurato che funziona come notebook Amazon SageMaker, in cui sono installati diversi comandi magic specifici per Neptune. Questi comandi magic semplificano le operazioni più comuni di Neptune, come il controllo dello stato del cluster, l'esecuzione degli attraversamenti SPARQL e Gremlin e l'esecuzione di un'operazione di caricamento in blocco.

Per avviare il processo di caricamento in blocco, usa il comando magic `%load`, che fornisce un'interfaccia per eseguire il Comando dello strumento di caricamento Neptune.

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Seleziona aws-neptune-blazegraph-to-neptune.
3. Scegli Apri notebook.
4. Nell'istanza in esecuzione di Jupyter, seleziona un notebook esistente o creane uno nuovo utilizzando il kernel Python 3.
5. Nel tuo notebook, apri una cella, inserisci `%load` ed esegui la cella.
6. Imposta i parametri per lo strumento di caricamento in blocco:

    a. Per Origine, inserisci la posizione di un file di origine da importare: `s3://{bucket_name}/{file_name}`.

    b. Per Formato, scegli il formato appropriato, che in questo esempio è `nquads`.

    c. Per Carica ARN, inserisci l'ARN per il ruolo `IAMBulkLoad` (queste informazioni si trovano nella console IAM all'interno di Ruoli).

7. Scegli Invia.

Il risultato contiene lo stato della richiesta. I caricamenti in blocco sono spesso processi che richiedono molto tempo, quindi la ricezione di una risposta non significa che il caricamento sia stato completato, ma solo che è iniziato. Queste informazioni sullo stato vengono aggiornate periodicamente fino a quando non viene segnalato che il processo è stato completato.

> **Note**
>
> Queste informazioni sono disponibili anche nel post del blog Moving to the cloud: Migrating Blazegraph to Amazon Neptune.

Caricamento di dati in Amazon Neptune

Esistono diversi modi per caricare i dati dei grafi in Amazon Neptune:

- Se è necessario caricare solo una quantità relativamente piccola di dati, è possibile utilizzare passaggi come istruzioni `INSERT` SPARQL o Gremlin `addV` e `addE`.
- È possibile trarre vantaggio da Strumento di caricamento in blocco Neptune per l'acquisizione di grandi quantità di dati che risiedono in file esterni. Il comando dello strumento di caricamento in blocco è più veloce e ha meno overhead rispetto ai comandi del linguaggio query. È ottimizzato per set di dati di grandi dimensioni e supporta sia i dati RDF (Resource Description Framework) che i dati Gremlin.
- È possibile utilizzare AWS Database Migration Service (AWS DMS) per importare dati da altri archivi dati (consultare Utilizzo AWS Database Migration Service per caricare dati in Amazon Neptune da un altro data store la Guida per AWS Database Migration Service l'utente).
- Infine, è possibile usare il passaggio `g.io(URL).read()` di Gremlin per leggere i file di dati in GraphML (un formato XML), GraphSon (un formato JSON) e altri formati. Consultate TinkerPopla documentazione per i dettagli.

Argomenti

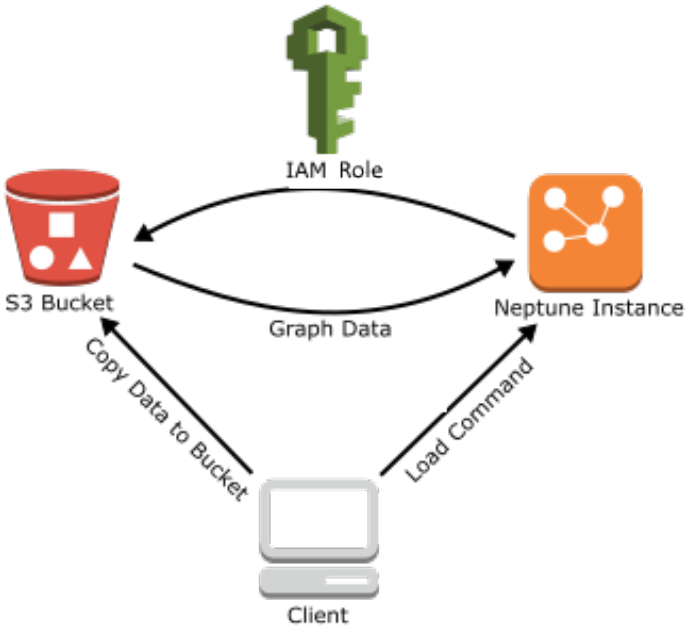
- Uso dello strumento di caricamento in blocco Amazon Neptune per importare i dati
- Utilizzo AWS Database Migration Service per caricare dati in Amazon Neptune da un altro data store

Uso dello strumento di caricamento in blocco Amazon Neptune per importare i dati

In Amazon Neptune è disponibile un comando `Loader` che consente di caricare dati da file esterni direttamente in un cluster database Neptune. Puoi utilizzare questo comando come alternativa all'esecuzione di un numero elevato di istruzioni `INSERT`, passaggi `addV` e `addE` o altre chiamate API.

Il comando Loader di Neptune è più rapido, meno oneroso, è ottimizzato per set di dati di grandi dimensioni e supporta sia i dati Gremlin che i dati RDF (Resource Description Framework) utilizzati da SPARQL.

Il seguente diagramma mostra una panoramica del processo di caricamento:



Di seguito sono descritte le fasi del processo di caricamento:

1. Copiare i file di dati in un bucket Amazon Simple Storage Service (Amazon S3).
2. Creare un ruolo IAM con accesso al bucket per la lettura e la presentazione di elenchi.
3. Creare un endpoint VPC Amazon S3.
4. Avviare strumento di caricamento Neptune inviando una richiesta tramite HTTP all'istanza database Neptune.
5. L'istanza database Neptune presuppone che il ruolo IAM carichi i dati dal bucket.

> **Note**
>
> È possibile caricare dati crittografati da Amazon S3 se sono stati crittografati utilizzando la modalità `SSE-S3` o `SSE-KMS` di Amazon S3, a condizione che il ruolo utilizzato per il caricamento in blocco abbia accesso all'oggetto Amazon S3 e anche, nel caso di SSE-KMS, a `kms:decrypt`. Neptune può quindi impersonare le credenziali dell'utente ed effettuare chiamate a `s3:getObject` per conto dell'utente.
> Tuttavia, attualmente Neptune non supporta il caricamento di dati crittografati utilizzando la modalità `SSE-C`.

Nelle seguenti sezioni vengono fornite istruzioni per la preparazione e il caricamento di dati in Neptune.

Argomenti

Prerequisiti: ruolo IAM e accesso ad Amazon S3

Il caricamento di dati da un bucket Amazon Simple Storage Service (Amazon S3) richiede AWS Identity and Access Management un ruolo (IAM) che abbia accesso al bucket. Tale ruolo viene assunto da Amazon Neptune per caricare i dati.

> **Note**
>
> È possibile caricare i dati crittografati da Amazon S3 se sono stati crittografati utilizzando la modalità `SSE-S3` di Amazon S3. In questo caso, Neptune è in grado di impersonare le credenziali dell'utente e di effettuare chiamate a `s3:getObject` per conto dell'utente.
> È possibile anche caricare i dati crittografati da Amazon S3 che sono stati crittografati utilizzando la modalità `SSE-KMS`, purché il ruolo IAM includa le autorizzazioni necessarie per accedere a AWS KMS. Senza AWS KMS le autorizzazioni appropriate, l'operazione di caricamento in blocco non riesce e restituisce una risposta. `LOAD_FAILED`
> Neptune non supporta attualmente il caricamento di dati Amazon S3 crittografati utilizzando la modalità `SSE-C`.

Le seguenti sezioni mostrano come utilizzare una policy IAM gestita per creare un ruolo IAM per l'accesso alle risorse Amazon S3 e quindi collegare il ruolo al cluster Neptune.

Argomenti

- Creazione dell'endpoint VPC Amazon S3
- Concatenazione di ruoli IAM in Amazon Neptune

> **Note**
>
> Per seguire le istruzioni devi disporre dell'accesso alla console IAM e delle autorizzazioni per gestire ruoli e policy IAM. Per ulteriori informazioni, consulta Autorizzazioni per l'utilizzo della console di AWS gestione nella Guida per l'utente IAM.
> La console di Amazon Neptune richiede che l'utente disponga delle seguenti autorizzazioni IAM per collegare il ruolo al cluster Neptune.
>
> ```
> iam:GetAccountSummary on resource: *
> iam:ListAccountAliases on resource: *
> iam:PassRole on resource: * with iam:PassedToService restricted to
>  rds.amazonaws.com
> ```

Creazione di un ruolo IAM per consentire ad Amazon Neptune di accedere alle risorse di Amazon S3

Utilizzare la policy IAM gestita `AmazonS3ReadOnlyAccess` per creare un nuovo ruolo IAM che consenta ad Amazon Neptune di accedere alle risorse di Amazon S3.

Per creare un nuovo ruolo IAM che consenta a Neptune di accedere ad Amazon S3

1. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, seleziona Roles (Ruoli).
3. Selezionare Create role (Crea ruolo).
4. In Servizio AWS , scegli S3.
5. Scegli Successivo: Autorizzazioni.
6. Utilizza la casella del filtro per filtrare in base al termine S3 e seleziona la casella accanto ad AmazonS3. ReadOnlyAccess

> **Note**
>
> Questa policy concede le autorizzazioni `s3:Get*` e `s3:List*` per tutti i bucket. Le fasi successive limitano l'accesso al ruolo utilizzando la policy di attendibilità.
> Lo strumento di caricamento richiede solo le autorizzazioni `s3:Get*` e `s3:List*` per il bucket da cui si esegue il caricamento, quindi è possibile limitare queste autorizzazioni anche in base alla risorsa Amazon S3.
> Se il bucket S3 è crittografato, occorre aggiungere autorizzazioni `kms:Decrypt`

7. Scegli Prossimo: Rivedi.
8. Impostare Nome ruolo su un nome per il ruolo IAM, ad esempio `NeptuneLoadFromS3`. È anche possibile aggiungere un valore opzionale in Descrizione ruolo, ad esempio "Consente a Neptune di accedere alle risorse di Amazon S3 per conto tuo".
9. Selezionare Crea ruolo.
10. Nel riquadro di navigazione, seleziona Ruoli.
11. Nel campo Search (Cerca) digita il nome del ruolo creato e selezionalo quando viene visualizzato nell'elenco.
12. Nella scheda Trust relationships (Relazioni di trust) scegli Edit trust relationship (Modifica relazione di trust).
13. Nel campo di testo incollare la seguente policy di attendibilità.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

14. Scegliere Update trust Policy (Aggiorna policy di attendibilità).

15. Completa le fasi descritte in Aggiunta del ruolo IAM a un cluster Amazon Neptune.

Aggiunta del ruolo IAM a un cluster Amazon Neptune

È possibile utilizzare la console per aggiungere il ruolo IAM a un cluster Amazon Neptune. In questo modo qualsiasi istanza database Neptune nel cluster può assumere il ruolo e caricare dati da Amazon S3.

> **Note**
>
> La console di Amazon Neptune richiede che l'utente disponga delle seguenti autorizzazioni IAM per collegare il ruolo al cluster Neptune.
>
> ```
> iam:GetAccountSummary on resource: *
> iam:ListAccountAliases on resource: *
> iam:PassRole on resource: * with iam:PassedToService restricted to
>  rds.amazonaws.com
> ```

Per aggiungere un ruolo IAM a un cluster Amazon Neptune

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Scegli il pulsante di opzione accanto al cluster che desideri modificare.
4. In Ruoli IAM, scegli Aggiungi ruolo.
5. In Aggiungi ruoli IAM a questo cluster (in Gestisci ruoli IAM), scegli il ruolo che hai creato nella sezione precedente.
6. Scegliere Add role (Aggiungi ruolo).
7. Attendi che il ruolo IAM diventi accessibile al cluster prima di utilizzarlo.

Creazione dell'endpoint VPC Amazon S3

Lo strumento di caricamento Neptune richiede un endpoint VPC di tipo Gateway per Amazon S3.

Per configurare l'accesso ad Amazon S3

1. Accedi AWS Management Console e apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nel pannello di navigazione, seleziona Endpoints (Endpoint).
3. Scegliere Create Endpoint (Crea endpoint).
4. Scegli il Nome servizio `com.amazonaws.region.s3` per l'endpoint di tipo Gateway.

   > **Note**
   >
   > Se la regione indicata qui non è corretta, verifica che lo sia la regione della console.

5. Scegli il VPC che contiene l'istanza database Neptune (è elencato per l'istanza database nella console Neptune).
6. Selezionare la casella di controllo accanto alle tabelle di routing associate alle sottoreti correlate al proprio cluster. Se si dispone di una sola tabella di routing, è necessario selezionare la casella corrispondente.
7. Scegliere Create Endpoint (Crea endpoint).

Per ulteriori informazioni sulla creazione dell'endpoint, consulta Endpoint VPC nella Guida per l'utente di Amazon VPC. Per informazioni sulle limitazioni degli endpoint VPC, consulta Endpoint VPC per Amazon S3.

Fasi successive

Ora che hai concesso l'accesso al bucket Amazon S3, puoi prepararti a caricare i dati. Per informazioni sui formati supportati, vedi Formati dei dati di caricamento.

Concatenazione di ruoli IAM in Amazon Neptune

> **Important**
>
> La nuova funzionalità di caricamento in blocco multi-account introdotta nel rilascio del motore 1.2.1.0.R3, che sfrutta il concatenamento dei ruoli IAM, può in alcuni casi causare un peggioramento delle prestazioni di caricamento in blocco. Di conseguenza, gli aggiornamenti ai rilasci del motore che supportano questa funzionalità sono stati temporaneamente sospesi fino alla risoluzione del problema.

Quando si collega un ruolo al cluster, il cluster può assumere tale ruolo per accedere ai dati archiviati in Amazon S3. A partire dal rilascio del motore 1.2.1.0.R3, se quel ruolo non ha accesso a tutte le risorse necessarie, è possibile concatenare uno o più ruoli aggiuntivi che il cluster può assumere per accedere ad altre risorse. Ogni ruolo nella catena assume il ruolo successivo nella catena, fino a quando il cluster non assume il ruolo alla fine della catena.

Per concatenare i ruoli, è possibile stabilire una relazione di trust tra di essi. Ad esempio, per concatenare `RoleB` a `RoleA`, `RoleA` deve disporre di una policy di autorizzazioni che gli consenta di assumere `RoleB` e `RoleB` deve disporre di una policy di trust che gli consenta di passare le proprie autorizzazioni a `RoleA`. Per ulteriori informazioni, consulta Utilizzo di ruoli IAM.

Il primo ruolo di una catena deve essere collegato al cluster che sta caricando i dati.

Il primo ruolo e ogni ruolo successivo che assume il ruolo seguente nella catena devono avere:

- Una policy che include una dichiarazione specifica con l'effetto `Allow` sull'azione `sts:AssumeRole`.
- Il nome della risorsa Amazon (ARN) del ruolo successivo in un elemento `Resource`.

> **Note**
>
> Il bucket Amazon S3 di destinazione deve trovarsi nella stessa AWS regione del cluster.

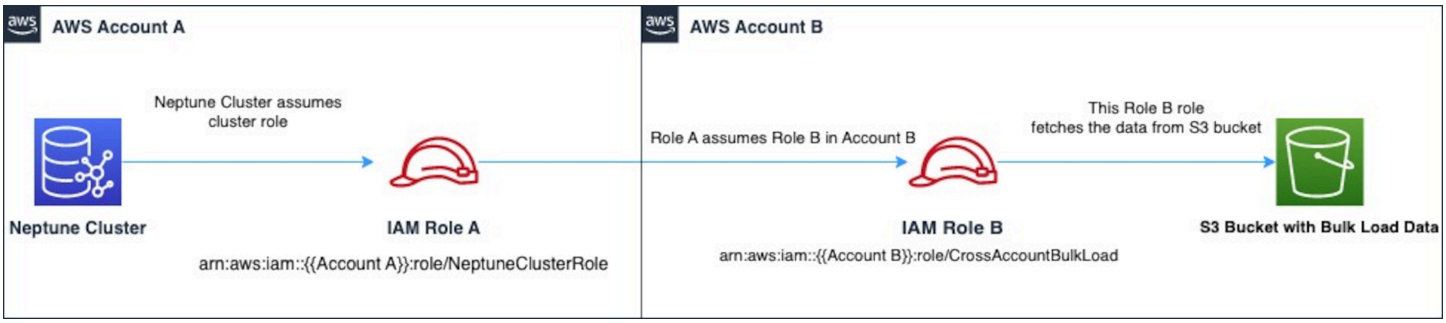
Accesso multi-account tramite ruoli concatenati

È possibile concedere l'accesso multi-account concatenando uno o più ruoli che appartengono a un altro account. Quando il cluster assume temporaneamente un ruolo appartenente a un altro account, può accedere alle relative risorse.

Supponiamo, ad esempio, che l'Account A voglia accedere ai dati in un bucket Amazon S3 che appartiene all'Account B:

- L'account A crea un ruolo AWS di servizio `RoleA` denominato per Neptune e lo collega a un cluster.
- L'Account B crea un ruolo denominato `RoleB` autorizzato ad accedere ai dati in un bucket dell'Account B.
- L'Account A collega una policy di autorizzazioni a `RoleA` che gli consente di assumere `RoleB`.

- L'Account B collega una policy di trust a `RoleB` che gli consente di passare le proprie autorizzazioni a `RoleA`.
- Per accedere ai dati nel bucket dell'Account B, l'Account A esegue un comando dello strumento di caricamento con un parametro `iamRoleArn` che concatena `RoleA` e `RoleB`. Per la durata dell'operazione loader, `RoleA` assume temporaneamente il ruolo `RoleB` per accedere al bucket Amazon S3 nell'Account B.



Ad esempio, `RoleA` avrà una policy di trust che stabilisce una relazione di trust con Neptune:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
          "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

`RoleA` avrà anche una policy di autorizzazione che gli consente di assumere il ruolo `RoleB`, che è di proprietà dell'Account B:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1487639602000",
      "Effect": "Allow",
      "Action": [
```

```
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::(Account B ID):role/RoleB"
    }
  ]
}
```

Al contrario, `RoleB` avrà una policy di trust per stabilire una relazione di trust con `RoleA`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "AWS": "arn:aws:iam::(Account A ID):role/RoleA"
      }
    }
  ]
}
```

`RoleB` avrà inoltre bisogno dell'autorizzazione per accedere ai dati nel bucket Amazon S3 situato nell'Account B.

Creazione di un AWS Security Token Service endpoint VPC (STS)

Il loader Neptune richiede un AWS STS endpoint VPC per quando concatenate i ruoli IAM per accedere privatamente alle API tramite indirizzi IP privati. AWS STS Puoi connetterti direttamente da un Amazon VPC a AWS STS un endpoint VPC in modo sicuro e scalabile. L'utilizzo di un endpoint VPC dell'interfaccia offre un migliore assetto di sicurezza in quanto non è necessario aprire firewall per il traffico in uscita. Offre inoltre gli altri vantaggi derivanti dall'utilizzo degli endpoint Amazon VPC.

Quando si utilizza un endpoint VPC, il traffico verso AWS STS non viene trasmesso su Internet e non esce mai dalla rete Amazon. Il tuo VPC è connesso in modo sicuro AWS STS senza rischi di disponibilità o vincoli di larghezza di banda sul traffico di rete. Per ulteriori informazioni, consulta Utilizzo di endpoint VPC dell'interfaccia AWS STS.

Per configurare l'accesso per (STS) AWS Security Token Service

1. Accedi AWS Management Console e apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.

2. Nel pannello di navigazione, seleziona Endpoints (Endpoint).
3. Scegliere Create Endpoint (Crea endpoint).
4. Scegli il Nome servizio `com.amazonaws.region.sts` per l'endpoint di tipo Interfaccia.
5. Scegli il VPC che contiene l'istanza database Neptune e l'istanza EC2.
6. Seleziona la casella di controllo accanto alla sottorete in cui è presente l'istanza EC2. Non è possibile selezionare più sottoreti dalla stessa zona di disponibilità.
7. Per IP address type (Tipo di indirizzo IP), seleziona una delle opzioni seguenti:
   - IPv4: consente di assegnare indirizzi IPv4 alle interfacce di rete dell'endpoint. Questa opzione è supportata solo se tutte le sottoreti selezionate dispongono di intervalli di indirizzi IPv4.
   - IPv6: consente di assegnare indirizzi IPv6 alle interfacce di rete dell'endpoint. Questa opzione è supportata solo se tutte le sottoreti selezionate sono sottoreti solo IPv6.
   - Dualstack: consente di assegnare sia indirizzi IPv4 che IPv6 alle interfacce di rete dell'endpoint. Questa opzione è supportata solo se tutte le sottoreti selezionate dispongono di intervalli di indirizzi IPv4 e IPv6.
8. Per Gruppi di sicurezza, seleziona i gruppi di sicurezza da associare alle interfacce di rete dell'endpoint per l'endpoint VPC. È necessario selezionare tutti i gruppi di sicurezza collegati all'istanza database Neptune e all'istanza EC2.
9. Per Policy, seleziona Full access (Accesso completo) per consentire tutte le operazioni da parte di tutti i principali su tutte le risorse dell'endpoint VPC. In caso contrario, seleziona Custom (Personalizza) per allegare una policy dell'endpoint VPC in grado di verificare le autorizzazioni di cui dispongono i principali per eseguire operazioni sulle risorse dell'endpoint VPC. Questa opzione è disponibile solo se il servizio supporta le policy dell'endpoint VPC. Per ulteriori informazioni, consulta Policy di endpoint.
10. (Facoltativo) Per aggiungere un tag, scegli Aggiungi nuovo tag e immetti la chiave e il valore del tag desiderati.
11. Seleziona Crea endpoint.

Per ulteriori informazioni sulla creazione dell'endpoint, consulta Endpoint VPC nella Guida per l'utente di Amazon VPC. Tieni presente che l'endpoint VPC di Amazon STS è un prerequisito obbligatorio per il concatenamento dei ruoli IAM.

Ora che hai concesso l'accesso all' AWS STS endpoint, puoi prepararti a caricare i dati. Per ulteriori informazioni sui formati di dati supportati, consulta Formati dei dati di caricamento.

Concatenamento dei ruoli all'interno di un comando dello strumento di caricamento

È possibile specificare il concatenamento dei ruoli quando si esegue un comando dello strumento di caricamento includendo un elenco separato da virgole di ARN di ruoli nel parametro `iamRoleArn`.

Sebbene nella maggior parte dei casi sia necessario avere solo due ruoli in una catena, è sicuramente possibile concatenarne tre o più. Ad esempio, questo comando dello strumento di caricamento concatena tre ruoli:

```
curl -X POST https://localhost:8182/loader \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://(the target bucket name)/(the target date file name)",
        "iamRoleArn" : "arn:aws:iam::(Account A ID):role/(RoleA),arn:aws:iam::(Account
 B ID):role/(RoleB),arn:aws:iam::(Account C ID):role/(RoleC)",
        "format" : "csv",
        "region" : "us-east-1"
      }'
```

Formati dei dati di caricamento

L'API Amazon Neptune `Load` supporta il caricamento di dati in una varietà di formati.

Formati di caricamento dei grafi di proprietà

È possibile eseguire query sui dati caricati in uno dei seguenti formati dei grafi di proprietà utilizzando sia Gremlin che openCypher:

- Formato dati di caricamento Gremlin (`csv`): formato di valori separati da virgole (CSV).
- Formato di caricamento dei dati openCypher (`opencypher`): formato di valori separati da virgole (CSV).

Formati di caricamento RDF

Per caricare i dati Resource Descrizione Framework (RDF) sui cui è possibile eseguire query utilizzando SPARQL, puoi utilizzare uno dei seguenti formati standard come specificato dal World Wide Web Consortium (W3C):

- N-Triples (`ntriples`) in base alle specifiche all'indirizzo https://www.w3.org/TR/n-triples/.

- N-Quads (`nquads`) in base alle specifiche all'indirizzo https://www.w3.org/TR/n-quads/.
- RDF/XML (`rdfxml`) in base alle specifiche all'indirizzo https://www.w3.org/TR/rdf-syntax-grammar/.
- Turtle (`turtle`) in base alle specifiche all'indirizzo https://www.w3.org/TR/turtle/.

I dati di caricamento devono utilizzare la codifica UTF-8

> **Important**
>
> Tutti i file di dati di caricamento devono essere codificati in formato UTF-8. Se un file non è codificato in formato UTF-8, Neptune prova comunque a caricarlo come UTF-8.

Per i dati N-Quads e N-triples che includono caratteri Unicode, sono supportate le sequenze di escape `\u`*`xxxxx`*. Tuttavia, Neptune non supporta la normalizzazione. Se è presente un valore che richiede la normalizzazione, non corrisponderà byte-to-byte durante l'interrogazione. Per ulteriori informazioni sulla normalizzazione, vedi la pagina relativa alla normalizzazione su Unicode.org.

Se i dati non sono in un formato supportato, è necessario convertirli prima di caricarli.

Uno strumento per convertire GraphML nel formato CSV di Neptune è disponibile nel progetto GraphML2CSV su. GitHub

Supporto della compressione per i file di dati di caricamento

Neptune supporta la compressione di singoli file in formato `gzip` o `bzip2`.

Il file compresso deve avere un'estensione `.gz` o `.bz2` e deve essere un singolo file di testo codificato in formato UTF-8. È possibile caricare più file, ma ognuno di essi deve essere un file `.gz`, `.bz2` o di testo non compresso separato. I file di archivio con estensioni come `.tar`, `.tar.gz` e `.tgz` non sono supportati.

Nelle seguenti sezioni vengono descritti i formati in modo più dettagliato.

Argomenti

- Formato dati di caricamento Gremlin
- Formato di caricamento dei dati openCypher
- Formati dei dati di caricamento RDF

Formato dati di caricamento Gremlin

Per caricare i dati di Apache TinkerPop Gremlin utilizzando il formato CSV, è necessario specificare i vertici e gli spigoli in file separati.

Lo strumento di caricamento è in grado di caricare da più file vertex e da più file edge in un singolo processo di caricamento.

Per ciascun comando di caricamento, il set di file da caricare deve trovarsi nella stessa cartella nel bucket Amazon S3 ed è necessario specificare il nome della cartella per il parametro `source`. I nomi file e le relative estensioni non sono importanti.

Il formato CSV Amazon Neptune segue la specifica CSV RFC 4180. Per ulteriori informazioni, vedi Common Format and MIME Type for CSV Files nel sito Web Internet Engineering Task Force (IETF).

> **Note**
>
> Tutti i file devono essere codificati in formato UTF-8.

Ogni file ha una riga di intestazione separata da virgola. La riga di intestazione è costituita da intestazioni di colonna di sistema e da intestazioni di colonna delle proprietà.

Intestazioni di colonna di sistema

Le intestazioni di colonna di sistema necessarie e consentite sono diverse per i file di vertice e i file di edge.

Ogni colonna di sistema può essere presente una sola volta in un'intestazione.

Per tutte le etichette viene fatta distinzione tra maiuscole e minuscole.

Intestazioni vertici

- `~id` - Campo obbligatorio

  Un ID per il vertice.
- `~label`

  Un'etichetta per il vertice. Sono consentiti più valori di etichetta, separati da punto e virgola (`;`).

Se non `~label` è presente, TinkerPop fornisce un'etichetta con il valore`vertex`, poiché ogni vertice deve avere almeno un'etichetta.

Intestazioni edge

- `~id` - Campo obbligatorio

  Un ID per l'edge.

- `~from` - Campo obbligatorio

  L'ID del vertice from.

- `~to` - Campo obbligatorio

  L'ID del vertice to.

- `~label`

  Un'etichetta per l'edge. Gli edge possono avere solo un'etichetta singola.

  Se non `~label` è presente, TinkerPop fornisce un'etichetta con il valore`edge`, perché ogni bordo deve avere un'etichetta.

Intestazioni di colonna delle proprietà

È possibile specificare una colonna (`:`) per una proprietà utilizzando la sintassi seguente. Per i nomi dei tipi non viene fatta distinzione tra maiuscole e minuscole. Tenere presente, tuttavia, che se all'interno del nome di una proprietà compaiono i due punti, è necessario che siano preceduti da una barra rovesciata: `\:`.

```
propertyname:type
```

> **Note**
>
> Lo spazio, la virgola, il ritorno a destra e i caratteri di nuova riga non sono consentiti nelle intestazioni delle colonne, pertanto i nomi delle proprietà non possono includere questi caratteri.

È possibile specificare una colonna per un tipo di matrice aggiungendo `[]` al tipo:

```
propertyname:type[]
```

> **Note**
>
> Le proprietà edge possono avere solo un singolo valore e verrà generato un errore se viene specificato un tipo di array o un secondo valore.

L'esempio seguente mostra l'intestazione di colonna per una proprietà denominata `age` del tipo `Int`.

```
age:Int
```

Ogni riga del file dovrà avere un valore intero in quella posizione o essere lasciata vuota.

Sono consentiti array di stringhe, ma le stringhe in un array non possono includere il carattere punto e virgola (`;`) a meno che non venga preceduto da una barra rovesciata (come in questo caso :`\;`).

Specifica la cardinalità di una colonna

A partire da Rilascio 1.0.1.0.200366.0 (26/07/2019), l'intestazione della colonna può essere utilizzata per specificare la cardinalità per la proprietà identificata dalla colonna. Ciò permette allo strumento di caricamento in blocco di rispettare la cardinalità allo stesso modo delle query Gremlin.

Per specificare la cardinalità di una colonna:

```
propertyname:type(cardinality)
```

Il valore *`cardinalità`* deve essere `single` o `set`. L'impostazione predefinita presunta è `set`; ciò significa che la colonna può accettare valori multipli. Nel caso di file edge, la cardinalità è sempre singola e specificare qualsiasi altra cardinalità fa sì che lo strumento di caricamento generi un'eccezione.

Se la cardinalità è `single`, lo strumento di caricamento genera un errore se un valore precedente è già presente quando viene caricato un valore oppure se vengono caricati più valori. Questo comportamento può essere sovrascritto in modo che un valore esistente venga sostituito quando un nuovo valore viene caricato usando il flag `updateSingleCardinalityProperties`. Per informazioni, consulta Comando dello strumento di caricamento.

È possibile utilizzare una cardinalità impostando un tipo di matrice, anche se non è generalmente necessario. Di seguito sono elencate le combinazioni possibili:

- `name:type`: la cardinalità è `set` e il contenuto è a valore singolo.
- `name:type[]`: la cardinalità è `set` e il contenuto ha valori multipli.
- `name:type(single)`: la cardinalità è `single` e il contenuto è a valore singolo.
- `name:type(set)`: la cardinalità è `set`, corrispondente al valore predefinito; il contenuto è a valore singolo.
- `name:type(set)[]`: la cardinalità è `set` e il contenuto ha valori multipli.
- `name:type(single)[]`: questa combinazione è contraddittoria e provoca un errore.

Nella sezione seguente sono elencati tutti i tipi di dati Gremlin disponibili.

Tipi di dati Gremlin

Si tratta di un elenco dei tipi di proprietà consentiti, con una descrizione per ognuno di essi.

Bool (o Booleano)

Indica un campo Booleano. Valori consentiti: `false`, `true`

> **Note**
>
> Qualsiasi valore diverso da `true` verrà trattato come false.

Tipi Numero intero

I valori al di fuori degli intervalli definiti determinano un errore.

Type	Intervallo
Byte	-128 to 127
Short	-32768 to 32767
Int	-2^31 to 2^31-1

Long	-2^63 to 2^63-1

Tipi Numero decimale

Supporta la notazione sia decimale sia scientifica. Inoltre, consente simboli come (+/-) Infinity o NaN. INF non è supportato.

Type	Intervallo
Float	32-bit IEEE 754 floating point
Double	64-bit IEEE 754 floating point

I valori float e doppi che sono troppo lunghi vengono caricati e arrotondati al valore più vicino per 24 bit (float) e 53 bit (doppi). Un valore medio viene arrotondato a 0 per le ultime cifre rimanenti a livello di bit.

Stringa

Le virgolette sono facoltative. Per virgole, nuova riga e ritorno a capo viene automaticamente inserito un carattere di escape, se sono inclusi in una stringa racchiusa tra virgolette doppie ("). Esempio: `"Hello, World"`

Per includere le virgolette in una stringa tra virgolette, è possibile inserire il carattere di escape per le virgolette utilizzandone due consecutivi: Esempio: `"Hello ""World"""`

Sono consentiti array di stringhe, ma le stringhe in un array non possono includere il carattere punto e virgola (`;`) a meno che non venga preceduto da una barra rovesciata (come in questo caso :`\;`).

Se desideri racchiudere le stringhe in una matrice con le virgolette, devi racchiudere l'intera matrice con una serie di virgolette. Esempio: `"String one; String 2; String 3"`

Data

Data Java i formato ISO-8601. Supporta i seguenti formati: `yyyy-MM-dd`, `yyyy-MM-ddTHH:mm`, `yyyy-MM-ddTHH:mm:ss`, `yyyy-MM-ddTHH:mm:ssZ`

Formato di riga Gremlin

Delimitatori

I campi in una riga sono separati da una virgola. I record sono separati da una nuova riga o da una nuova riga seguita da un ritorno a capo.

Campi vuoti

I campi vuoti sono consentiti per le colonne non obbligatorie (ad esempio le proprietà definite dall'utente). Un campo vuoto richiede comunque un separatore virgola. L'esempio nella sezione successiva presenta un campo vuoto in ciascun vertice.

ID vertici

I valori `~id` devono essere univoci per tutti i vertici in ogni file di vertici. A un singolo vertice nel grafo vengono applicate più righe di vertice con valori `~id` identici.

ID edge

I valori `~id` devono essere univoci anche per tutti gli edge in ogni file di edge. All'unico edge nel grafo vengono applicate più righe di edge con valori `~id` identici.

Etichette

Per le etichette viene fatta distinzione tra maiuscole e minuscole.

Valori delle stringhe

Le virgolette sono facoltative. Per virgole, nuova riga e ritorno a capo viene automaticamente inserito un carattere di escape, se sono inclusi in una stringa racchiusa tra virgolette doppie (`''`).

Specifica del formato CSV

Il formato CSV Neptune segue le specifiche CSV RFC 4180, inclusi i seguenti requisiti.

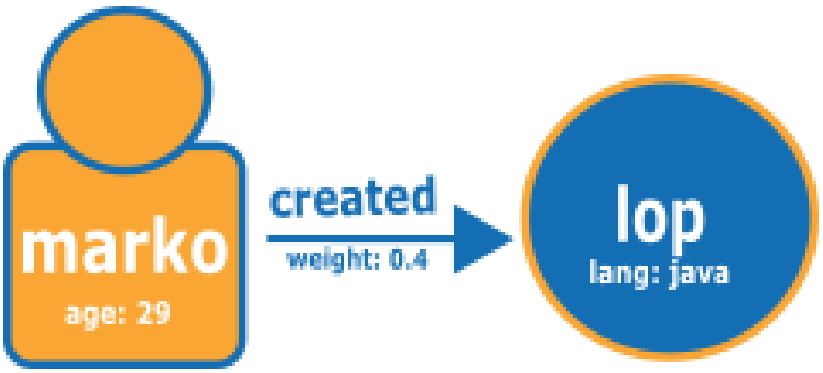
- Sono supportate le terminazioni di riga di entrambi gli stili, Unix e Windows (\ n o\r\n).
- Qualsiasi campo può essere racchiuso tra virgolette (doppie).
- I campi contenenti un'interruzione di riga, virgolette doppie o virgole devono essere racchiusi tra virgolette (in caso contrario, il caricamento viene interrotto immediatamente).
- Un carattere virgolette doppie (`''`) in un campo deve essere rappresentato da due (doppi) caratteri. Ad esempio, una stringa `Hello ''World''` deve essere presente nei dati come `''Hello ""World""''`.

- Gli spazi circostanti tra i delimitatori vengono ignorati. Se una riga è presente come `value1`, `value2`, vengono memorizzati come `"value1"` e `"value2"`.
- Tutti gli altri caratteri di escape vengono archiviati integralmente. Ad esempio, `"data1\tdata2"` viene archiviato come `"data1\tdata2"`. Se i caratteri sono racchiusi tra virgolette, non sono necessari ulteriori caratteri di escape.
- Sono consentiti campi vuoti. Un campo vuoto viene considerato come un valore vuoto.
- Per separare più valori per un campo viene utilizzato il punto e virgola (`;`).

Per ulteriori informazioni, vedi Common Format and MIME Type for CSV Files nel sito Web Internet Engineering Task Force (IETF).

Esempio Gremlin

Il diagramma seguente mostra un esempio di due vertici e uno spigolo presi dal Modern Graph. TinkerPop



Di seguito è riportato il grafo nel formato di caricamento CSV Neptune.

File del vertice:

```
~id, name:String, age:Int, lang:String, interests:String[], ~label
v1, "marko", 29, , "sailing;graphs", person
v2, "lop", , "java", , software
```

Vista tabulare del file del vertice:

~id	name:String	age:Int	lang:String	interests :String[]	~label

v1	"marko"	29		["sailing", "graphs"]	person
v2	"lop"		"java"		software

File di edge:

```
~id, ~from, ~to, ~label, weight:Double
e1, v1, v2, created, 0.4
```

Vista tabulare del file di edge:

~id	~from	~to	~label	weight:Double
e1	v1	v2	created	0.4

Fasi successive

Ora che hai una maggiore conoscenza dei formati di caricamento, vedi Esempio: caricamento di dati in un'istanza database Neptune.

Formato di caricamento dei dati openCypher

Per caricare i dati openCypher utilizzando il formato CSV openCypher, è necessario specificare nodi e relazioni in file separati. Lo strumento di caricamento può caricare da più di questi file di nodi e file di relazioni in un unico processo di caricamento.

Per ogni comando di caricamento, il set di file da caricare deve avere lo stesso prefisso di percorso in un bucket Amazon Simple Storage Service. Il prefisso viene specificato nel parametro di origine. I nomi e le estensioni effettivi dei file non sono importanti.

In Amazon Neptune, il formato CSV openCypher è conforme alla specifica CSV RFC 4180. Per ulteriori informazioni, vedi Common Format and MIME Type for CSV Files (https://tools.ietf.org/html/rfc4180) nel sito Web Internet Engineering Task Force (IETF).

> **Note**
>
> Questi file DEVONO essere codificati in formato UTF-8.

Ogni file ha una riga di intestazione separata da virgole che contiene sia intestazioni di colonna di sistema che intestazioni di colonna di proprietà.

Intestazioni di colonna di sistema nei file di caricamento dati openCypher

Una determinata colonna di sistema può essere presente una sola volta in ogni file. Tutte le etichette delle intestazioni di colonna di sistema fanno distinzione tra maiuscole e minuscole.

Le intestazioni di colonna di sistema obbligatorie e consentite sono diverse per i file di caricamento dei nodi e i file di caricamento delle relazioni openCypher:

Intestazioni di colonna di sistema nei file dei nodi

- **`:ID`**: (obbligatorio) ID per il nodo.

  È possibile aggiungere uno spazio ID opzionale all'intestazione di colonna `:ID` del nodo in questo modo: `:ID(`*`ID Space`*`)`. Un esempio è `:ID(movies)`.

  Quando carichi le relazioni che collegano i nodi in questo file, usa gli stessi spazi ID nelle colonne `:START_ID` e/o `:END_ID` dei file delle relazioni.

  La colonna `:ID` del nodo può essere facoltativamente archiviata come proprietà nel formato *`property name`*`:ID`. Un esempio è `name:ID`.

  Gli ID nodo devono essere univoci in tutti i file dei nodi nei caricamenti correnti e precedenti. Se viene utilizzato uno spazio ID, gli ID nodo devono essere univoci per tutti i file dei nodi che utilizzano lo stesso spazio ID nei caricamenti correnti e precedenti.
- **`:LABEL`**: etichetta per il nodo.

  Sono consentiti più valori di etichetta, separati da punto e virgola (`;`).

Intestazioni di colonna di sistema nei file delle relazioni

- **`:ID`**: ID della relazione. È obbligatorio quando `userProvidedEdgeIds` è true (impostazione predefinita), ma non è valido quando `userProvidedEdgeIds` è `false`.

  Gli ID relazione devono essere univoci in tutti i file delle relazioni nei caricamenti correnti e precedenti.
- **`:START_ID`**: (obbligatorio) ID del nodo da cui inizia questa relazione.

Facoltativamente, è possibile associare uno spazio ID alla colonna ID iniziale nel formato `:START_ID(`*`ID Space`*`)`. Lo spazio ID assegnato all'ID nodo iniziale deve corrispondere allo spazio ID assegnato al nodo nel relativo file dei nodi.

- **`:END_ID`**: (obbligatorio) ID del nodo in cui termina questa relazione.

  Facoltativamente, è possibile associare uno spazio ID alla colonna ID finale nel formato `:END_ID(`*`ID Space`*`)`. Lo spazio ID assegnato all'ID nodo finale deve corrispondere allo spazio ID assegnato al nodo nel relativo file dei nodi.
- **`:TYPE`**: tipo della relazione. Le relazioni possono avere un solo tipo.

> **Note**
>
> Per informazioni su come vengono gestiti gli ID nodo o relazione duplicati dal processo di caricamento in blocco, vedi Caricamento di dati openCypher.

Intestazioni di colonna di proprietà nei file di caricamento dati openCypher

È possibile specificare che una colonna contenga i valori di una particolare proprietà utilizzando l'intestazione di una colonna di proprietà nel seguente formato:

```
propertyname:type
```

Lo spazio, la virgola, il ritorno a capo e i caratteri di nuova riga non sono consentiti nelle intestazioni delle colonne, pertanto i nomi delle proprietà non possono includere questi caratteri. Ecco un esempio di intestazione di colonna per una proprietà denominata `age` di tipo `Int`:

```
age:Int
```

La colonna con `age:Int` come intestazione di colonna dovrebbe quindi contenere un numero intero o un valore vuoto in ogni riga.

Tipi di dati nei file di caricamento dati di Neptune openCypher

- **`Bool`** o **`Boolean`**: campo booleano. I valori consentiti sono `true` e `false`.

  Qualsiasi valore diverso da `true` viene trattato come `false`.

- **`Byte`**: numero intero compreso tra `-128` e `127`.
- **`Short`**: numero intero compreso tra `-32,768` e `32,767`.
- **`Int`**: numero intero compreso tra `-2^31` e `2^31 - 1`.
- **`Long`**: numero intero compreso tra `-2^63` e `2^63 - 1`.
- **`Float`**: numero in virgola mobile IEEE 754 a 32 bit. Sono supportate sia la notazione decimale che la notazione scientifica. `Infinity`, `-Infinity` e `NaN` sono tutti riconosciuti, mentre `INF` non lo è.

  I valori che contengono troppe cifre vengono arrotondati al valore più vicino (un valore intermedio viene arrotondato a 0 per l'ultima cifra rimanente a livello di bit).
- **`Double`**: numero in virgola mobile IEEE 754 a 64 bit. Sono supportate sia la notazione decimale che la notazione scientifica. `Infinity`, `-Infinity` e `NaN` sono tutti riconosciuti, mentre `INF` non lo è.

  I valori che contengono troppe cifre vengono arrotondati al valore più vicino (un valore intermedio viene arrotondato a 0 per l'ultima cifra rimanente a livello di bit).
- **`String`**: le virgolette sono facoltative. Per i caratteri di virgola, nuova riga e ritorno a capo viene automaticamente inserito un carattere di escape, se sono inclusi in una stringa racchiusa tra virgolette doppie (") come `"Hello, World"`.

  È possibile includere le virgolette in una stringa tra virgolette usandone due di seguito come `"Hello ""World"""`.
- **`DateTime`**: data Java in uno dei seguenti formati ISO-8601:
  - `yyyy-MM-dd`
  - `yyyy-MM-ddTHH:mm`
  - `yyyy-MM-ddTHH:mm:ss`
  - `yyyy-MM-ddTHH:mm:ssZ`

Tipi di dati con trasmissione automatica nei file di caricamento dati di Neptune openCypher

I tipi di dati con trasmissione automatica vengono forniti per caricare tipi di dati attualmente non supportati in modo nativo da Neptune. I dati in tali colonne vengono archiviati letteralmente come stringhe senza alcuna verifica rispetto ai formati previsti. Sono consentiti i seguenti tipi di dati con trasmissione automatica:

- **`Char`**: campo `Char`. Archiviato come stringa.

- **`Date`**, **`LocalDate`** e **`LocalDateTime`**: vedi Neo4j Temporal Instants per una descrizione dei tipi `date`, `localdate` e `localdatetime`. I valori vengono caricati letteralmente come stringhe, senza convalida.
- **`Duration`**: vedi Neo4j Duration format. I valori vengono caricati letteralmente come stringhe, senza convalida.
- Point: campo Point per l'archiviazione di dati spaziali. Vedi Spatial instants. I valori vengono caricati letteralmente come stringhe, senza convalida.

Esempio del formato di caricamento openCypher

Il seguente diagramma tratto dal TinkerPop Modern Graph mostra un esempio di due nodi e una relazione:

Di seguito è riportato il grafo nel normale formato di caricamento di Neptune openCypher.

File dei nodi:

```
:ID,  name:String,  age:Int,  lang:String,  :LABEL
 v1,   "marko",      29,           ,        person
 v2,   "lop",          ,      "java",       software
```

File di relazione:

```
:ID,  :START_ID,  :END_ID,   :TYPE,      weight:Double
 e1,   v1,         v2,        created,    0.4
```

In alternativa, è possibile utilizzare gli spazi ID e l'ID come proprietà, come segue:

File del primo nodo:

```
name:ID(person),    age:Int,  lang:String,  :LABEL
 "marko",            29,           ,       person
```

File del secondo nodo:

```
name:ID(software),  age:Int,  lang:String,  :LABEL
 "lop",               ,        "java",       software
```

File di relazione:

```
:ID,  :START_ID,  :END_ID,   :TYPE,      weight:Double
 e1,   "marko",    "lop",     created,    0.4
```

Formati dei dati di caricamento RDF

Per caricare i dati Resource Descrizione Framework (RDF), puoi utilizzare uno dei seguenti formati standard come specificato dal World Wide Web Consortium (W3C):

- N-Triples (`ntriples`) in base alle specifiche all'indirizzo https://www.w3.org/TR/n-triples/
- N-Quads (`nquads`) in base alle specifiche all'indirizzo https://www.w3.org/TR/n-quads/
- RDF/XML (`rdfxml`) in base alle specifiche all'indirizzo https://www.w3.org/TR/rdf-syntax-grammar/
- Turtle (`turtle`) in base alle specifiche all'indirizzo https://www.w3.org/TR/turtle/

> **Important**
>
> Tutti i file devono essere codificati in formato UTF-8.
> Per i dati N-Quads e N-triples che includono caratteri Unicode, sono supportate le sequenze di escape `\u`*`xxxxx`*. Tuttavia, Neptune non supporta la normalizzazione. Se è presente un valore che richiede la normalizzazione, non corrisponderà byte-to-byte durante l'interrogazione. Per ulteriori informazioni sulla normalizzazione, vedi la pagina relativa alla normalizzazione su Unicode.org.

Fasi successive

Ora che hai una maggiore conoscenza dei formati di caricamento, vedi Esempio: caricamento di dati in un'istanza database Neptune.

Esempio: caricamento di dati in un'istanza database Neptune

Questo esempio illustra come caricare dati in Amazon Neptune. Se non diversamente specificato, è necessario seguire questi passaggi da un'istanza Amazon Elastic Compute Cloud (Amazon EC2) nello stesso Amazon Virtual Private Cloud (VPC) dell'istanza database Neptune.

Prerequisiti per l'esempio di caricamento di dati

Prima di iniziare, devi disporre di quanto segue:

- Istanza database Neptune.

  Per informazioni sull'avvio di un'istanza database Neptune, consulta Creazione di un nuovo cluster database Neptune.
- Bucket Amazon Simple Storage Service (Amazon S3) in cui inserire i file di dati.

  Puoi utilizzare un bucket esistente. Se non disponi di un bucket S3, consulta Creare un bucket nella Guida alle operazioni di base di Amazon S3.
- Dati del grafo da caricare, in uno dei formati supportati dallo strumento di caricamento Neptune:

  Se state usando Gremlin per interrogare il vostro grafico, Neptune può caricare i dati in un formato comma-separated-values (CSV), come descritto in. Formato dati di caricamento Gremlin

  Se si utilizza openCypher per eseguire query sul grafo, Neptune può anche caricare i dati in un formato CSV specifico di openCypher, come descritto in Formato di caricamento dei dati openCypher.

  Se si utilizza SPARQL, Neptune può caricare i dati in diversi formati RDF, come descritto in Formati dei dati di caricamento RDF.
- Ruolo IAM che deve essere assunto dall'istanza database Neptune dotato di una policy IAM che consenta l'accesso ai file di dati nel bucket S3. La policy deve concedere le autorizzazioni per la lettura e la presentazione di elenchi.

  Per informazioni su come creare un ruolo che abbia accesso ad Amazon S3 e su come associarlo a un cluster Neptune, vedi Prerequisiti: ruolo IAM e accesso ad Amazon S3.

> **Note**
>
> L'API Neptune `Load` necessita solo dell'accesso in lettura ai file di dati. Non è necessario che la policy IAM conceda l'accesso in scrittura o l'accesso all'intero bucket.

- Endpoint VPC Amazon S3. Per ulteriori informazioni, consulta la sezione Creazione di un endpoint VPC Amazon S3.

Creazione di un endpoint VPC Amazon S3

Lo strumento di caricamento Neptune richiede un endpoint VPC per Amazon S3.

Per configurare l'accesso ad Amazon S3

1. Accedi AWS Management Console e apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nel riquadro di navigazione a sinistra, scegli Endpoints (Endpoint).
3. Scegliere Create Endpoint (Crea endpoint).
4. Scegli il Service Name (Nome servizio) `com.amazonaws.region.s3`.

   > **Note**
   >
   > Se la regione indicata qui non è corretta, verifica che lo sia la regione della console.

5. Scegli il VPC che contiene l'istanza database Neptune.
6. Selezionare la casella di controllo accanto alle tabelle di routing associate alle sottoreti correlate al proprio cluster. Se si dispone di una sola tabella di routing, è necessario selezionare la casella corrispondente.
7. Scegliere Create Endpoint (Crea endpoint).

Per ulteriori informazioni sulla creazione dell'endpoint, consulta Endpoint VPC nella Guida per l'utente di Amazon VPC. Per informazioni sulle limitazioni degli endpoint VPC, consulta Endpoint VPC per Amazon S3.

Per caricare i dati in un'istanza database Neptune

1. Copiare i file di dati in un bucket Amazon S3. Il bucket S3 deve trovarsi nella stessa AWS regione del cluster che carica i dati.

   È possibile utilizzare il seguente AWS CLI comando per copiare i file nel bucket.

   > **Note**
   >
   > Non è necessario eseguire questo comando dall'istanza Amazon EC2.

   ```
   aws s3 cp data-file-name s3://bucket-name/object-key-name
   ```

   > **Note**
   >
   > In Amazon S3 un nome chiave dell'oggetto è il percorso intero di un file, incluso il nome. Esempio: nel comando `aws s3 cp datafile.txt s3://examplebucket/mydirectory/datafile.txt`, il nome chiave dell'oggetto è **`mydirectory/datafile.txt`**.

   In alternativa, puoi utilizzare il AWS Management Console per caricare i file nel bucket S3. Apri la console Amazon S3 all'indirizzo https://console.aws.amazon.com/s3/ e scegli un bucket. Nell'angolo superiore sinistro, scegli Upload (Carica) per caricare i file.

2. Da una finestra a riga di comando, inserisci quanto segue per eseguire lo strumento di caricamento Neptune, utilizzando i valori corretti per l'endpoint, il percorso Amazon S3, il formato e l'ARN del ruolo IAM.

   Il parametro `format` può essere uno dei seguenti valori: `csv` per Gremlin, `opencypher` per openCypher o `ntriples`, `nquads`, `turtle` e `rdfxml` per RDF. Per informazioni sugli altri parametri, vedi Comando dello strumento di caricamento Neptune.

   Per informazioni su come trovare il nome host dell'istanza database Neptune, consulta la sezione Connessione agli endpoint Amazon Neptune.

   Il parametro della regione deve corrispondere alla regione del cluster e del bucket S3.

Amazon Neptune è disponibile nelle seguenti regioni: AWS

- Stati Uniti orientali (Virginia settentrionale): `us-east-1`
- Stati Uniti orientali (Ohio): `us-east-2`
- Stati Uniti occidentali (California settentrionale): `us-west-1`
- Stati Uniti occidentali (Oregon): `us-west-2`
- Canada (Centrale): `ca-central-1`
- Sud America (San Paolo): `sa-east-1`
- Europa (Stoccolma): `eu-north-1`
- Europa (Irlanda): `eu-west-1`
- Europa (Londra): `eu-west-2`
- Europa (Parigi): `eu-west-3`
- Europa (Francoforte): `eu-central-1`
- Medio Oriente (Bahrein): `me-south-1`
- Medio Oriente (Emirati Arabi Uniti): `me-central-1`
- Israele (Tel Aviv): `il-central-1`
- Africa (Città del Capo): `af-south-1`
- Asia Pacifico (Hong Kong): `ap-east-1`
- Asia Pacifico (Tokyo): `ap-northeast-1`
- Asia Pacifico (Seoul): `ap-northeast-2`
- Asia Pacifico (Osaka): `ap-northeast-3`
- Asia Pacifico (Singapore): `ap-southeast-1`
- Asia Pacifico (Sydney): `ap-southeast-2`
- Asia Pacifico (Mumbai): `ap-south-1`
- Cina (Pechino): `cn-north-1`
- Cina (Ningxia): `cn-northwest-1`
- AWS GovCloud (Stati Uniti occidentali): `us-gov-west-1`
- AWS GovCloud (Stati Uniti orientali): `us-gov-east-1`

```
curl -X POST \
```

```
    -H 'Content-Type: application/json' \
    https://your-neptune-endpoint:port/loader -d '
    {
      "source" : "s3://bucket-name/object-key-name",
      "format" : "format",
      "iamRoleArn" : "arn:aws:iam::account-id:role/role-name",
      "region" : "region",
      "failOnError" : "FALSE",
      "parallelism" : "MEDIUM",
      "updateSingleCardinalityProperties" : "FALSE",
      "queueRequest" : "TRUE",
      "dependencies" : ["load_A_id", "load_B_id"]
    }'
```

Per informazioni su come creare un ruolo IAM e associarlo a un cluster Neptune, vedi Prerequisiti: ruolo IAM e accesso ad Amazon S3.

Note

Consulta Parametri della richiesta dello dello strumento di caricamento Neptune per informazioni dettagliate sui parametri della richiesta di caricamento. In breve:
Il parametro `source` accetta un URI Amazon S3 che punta a un file singolo o a una cartella. Se si specifica una cartella, Neptune carica tutti i file di dati presenti nella cartella.
La cartella può contenere più file vertex e più file edge.
l'URI può essere in uno dei seguenti formati.

- `s3://bucket_name/object-key-name`
- `https://s3.amazonaws.com/bucket_name/object-key-name`
- `https://s3-us-east-1.amazonaws.com/bucket_name/object-key-name`

Il parametro `format` può essere uno dei seguenti:

- Formato CSV Gremlin (`csv`) per i grafi di proprietà Gremlin
- Formato CSV openCypher (`opencypher`) per i grafi di proprietà openCypher
- Formato N -Triples (`ntriples`) per RDF/SPARQL
- Formato N-Quads (`nquads`) per RDF/SPARQL
- Formato RDF/XML (`rdfxml`) per RDF/SPARQL

- Formato Turtle (`turtle`) per RDF/SPARQL

Il parametro opzionale `parallelism` consente di limitare il numero di thread utilizzati nel processo di caricamento in blocco. Può essere impostato su `LOW`, `MEDIUM`, `HIGH` o `OVERSUBSCRIBE`.
Quando `updateSingleCardinalityProperties` è impostato su `"FALSE"`, lo strumento di caricamento restituisce un errore se viene fornito più di un valore in un file di origine caricato per una proprietà vertice a cardinalità singola o edge.
L'impostazione di `queueRequest` su `"TRUE"` fa sì che la richiesta di caricamento venga inserita in una coda se è già in esecuzione un'attività di caricamento.
Il parametro `dependencies` rende l'esecuzione della richiesta di caricamento subordinata al completamento di una o più attività di caricamento già inserita nella coda.

3. Lo strumento di caricamento Neptune restituisce un oggetto `id` del processo che consente di controllare lo stato o di annullare il processo di caricamento, ad esempio:

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

4. Inserisci il comando seguente per ottenere lo stato del caricamento con il valore `loadId` della Fase 3:

```
curl -G 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5'
```

Se lo stato del caricamento elenca un errore, è possibile richiedere uno stato più dettagliato e un elenco degli errori. Per maggiori informazioni ed esempi, consulta API Get-Status dello strumento di caricamento Neptune.

5. (Facoltativo) Annullare il processo `Load`.

inserisci il comando seguente per eseguire un `Delete` del processo dello strumento di caricamento con il valore `id` del processo della Fase 3:

```
curl -X DELETE 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5'
```

Il comando `DELETE` restituisce il codice HTTP `200 OK` ad annullamento eseguito.

I dati provenienti dai file dal processo di caricamento terminato non vengono ripristinati. I dati rimangono nell'istanza database Neptune.

Ottimizzazione di un caricamento in blocco di Amazon Neptune

Utilizzare le seguenti strategie per ridurre al minimo il tempo di caricamento per un caricamento in blocco di Neptune:

- Pulire i dati:
  - Assicurarsi di convertire i dati in un formato di dati supportato prima di caricarli.
  - Rimuovere eventuali duplicati o errori noti.
  - Ridurre il più possibile il numero di predicati univoci (come le proprietà di archi e vertici).
- Ottimizzare i file:
  - Se si caricano file di grandi dimensioni come i file CSV da un bucket Amazon S3, lo strumento di caricamento gestisce la concorrenza per l'utente analizzandoli in blocchi che possono essere caricati in parallelo. L'utilizzo di un numero molto elevato di file di piccole dimensioni può rallentare questo processo.
  - Se si caricano più file da una cartella Amazon S3, lo strumento di caricamento carica automaticamente prima i file dei vertici, poi quelli degli archi.
  - La compressione dei file riduce i tempi di trasferimento. Lo dello strumento di caricamento supporta la compressione `gzip` dei file di origine.
- Controllare le impostazioni dello dello strumento di caricamento:
  - Se non è necessario eseguire altre operazioni durante il caricamento, utilizzare il parametro `OVERSUBSCRIBE parallelism`. Questa impostazione dei parametri fa sì che lo strumento di caricamento in blocco utilizzi tutte le risorse della CPU disponibili durante l'esecuzione. In genere è necessario il 60-70% della capacità della CPU per mantenere l'operazione in esecuzione alla velocità consentita dai vincoli di I/O.

> **Note**
>
> Quando `parallelism` è impostato su `OVERSUBSCRIBE` o `HIGH` (l'impostazione predefinita), durante il caricamento dei dati openCypher c'è il rischio che i thread incontrino una race condition e un deadlock, causando un errore `LOAD_DATA_DEADLOCK`. In questo caso, impostare `parallelism` su un'impostazione inferiore e riprovare a eseguire il caricamento.

- Se il processo di caricamento include più richieste di caricamento, utilizzare il parametro `queueRequest`. L'impostazione di `queueRequest` su `TRUE` consente a Neptune di mettere in coda le richieste in modo da non dover aspettare che una finisca prima di inviarne un'altra.
- Se le richieste di caricamento vengono messe in coda, è possibile impostare i livelli di dipendenza utilizzando il parametro `dependencies`, in modo che l'esito negativo di un processo causi l'esito negativo dei processi dipendenti. In questo modo è possibile evitare incongruenze nei dati caricati.
- Se un processo di caricamento prevede l'aggiornamento di valori caricati in precedenza, assicurarsi di impostare il parametro `updateSingleCardinalityProperties` su `TRUE`. In caso contrario, lo strumento di caricamento considererà un errore il tentativo di aggiornare un valore di cardinalità singola esistente. Per i dati Gremlin, la cardinalità è specificata anche nelle intestazioni di colonna di proprietà (vedi Intestazioni di colonna delle proprietà).

  > **Note**
  >
  > Il parametro `updateSingleCardinalityProperties` non è disponibile per i dati RDF (Resource Description Framework).

- È possibile utilizzare il parametro `failOnError` per determinare se le operazioni di caricamento in blocco devono avere esito negativo o continuare quando si verifica un errore. Inoltre, è possibile utilizzare il parametro `mode` per assicurarsi che un processo di caricamento riprenda il caricamento dal punto in cui un processo precedente ha avuto esito negativo anziché ricaricare i dati che erano già stati caricati.

- Aumentare la capacità: impostare l'istanza di scrittura del cluster database sulla dimensione massima prima del caricamento in blocco. Tenere presente che se si esegue questa operazione, è necessario aumentare la capacità anche di tutte le istanze di replica di lettura nel cluster database o rimuoverle fino al termine del caricamento dei dati.

Una volta completato il caricamento in blocco, assicurarsi di dimensionare nuovamente l'istanza di scrittura.

> **Important**
>
> Se si verifica un ciclo di riavvii ripetuti delle repliche di lettura a causa del ritardo di replica durante un caricamento in blocco, è probabile che le repliche non riescano a stare al passo con l'istanza di scrittura nel cluster database. Dimensionare le istanze di lettura in modo che siano più grandi dell'istanza di scrittura oppure rimuoverle temporaneamente durante il caricamento in blocco e quindi ricrearle al termine dell'operazione.

Vedi Parametri della richiesta per maggiori dettagli sull'impostazione dei parametri di richiesta dello strumento di caricamento.

Documentazione di riferimento dello strumento di caricamento Neptune

In questa sezione vengono descritte le API `Loader` per Amazon Neptune disponibili dall'endpoint HTTP di un'istanza database Neptune.

> **Note**
>
> Vedi Messaggi di feed e di errore dello strumento di caricamento Neptune per un elenco dei messaggi di errore e di feed restituiti dallo strumento di caricamento in caso di errori.

Indice

Comando dello strumento di caricamento Neptune

Carica i dati da un bucket Amazon S3 in un'istanza database Neptune.

Per caricare i dati è necessario inviare una richiesta HTTP `POST` all'endpoint `https://`*`your-neptune-endpoint`*`:`*`port`*`/loader`. I parametri per la richiesta `loader` possono essere inviati nel corpo `POST` corpo o come parametri di codifica URL.

> **Important**
>
> Il tipo MIME deve essere `application/json`.

Il bucket S3 deve trovarsi nella stessa AWS regione del cluster.

> **Note**
>
> È possibile caricare i dati crittografati da Amazon S3 se sono stati crittografati utilizzando la modalità `SSE-S3` di Amazon S3. In questo caso, Neptune è in grado di impersonare le credenziali dell'utente e di effettuare chiamate a `s3:getObject` per conto dell'utente.
> È possibile anche caricare i dati crittografati da Amazon S3 che sono stati crittografati utilizzando la modalità `SSE-KMS`, purché il ruolo IAM includa le autorizzazioni necessarie per accedere a AWS KMS. Senza AWS KMS le autorizzazioni appropriate, l'operazione di caricamento in blocco non riesce e restituisce una risposta. `LOAD_FAILED`
> Neptune non supporta attualmente il caricamento di dati Amazon S3 crittografati utilizzando la modalità `SSE-C`.

Non è necessario attendere il completamento di un processo di caricamento prima di iniziarne un altro. Neptune può accodare fino a 64 richieste di processo alla volta, a condizione che i relativi parametri `queueRequest` siano tutti impostati su `"TRUE"`. L'ordine di coda dei lavori sarà first-in-first-out (FIFO). Se invece non si desidera che un processo di caricamento sia messo in coda, è possibile impostare il relativo parametro `queueRequest` su `"FALSE"` (valore predefinito), in modo che il caricamento abbia esito negativo se ne è già in corso un altro.

È possibile utilizzare il parametro `dependencies` per accodare un'attività che deve essere eseguita solo dopo che le attività precedenti specificate nella coda sono state completate correttamente. Se si esegue questa operazione e uno qualsiasi di queste attività specificate non riesce, l'attività non verrà eseguita e il relativo stato verrà impostato su `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

Sintassi della richiesta dello strumento di caricamento Neptune

```
{
  "source" : "string",
  "format" : "string",
  "iamRoleArn" : "string",
```

```
  "mode": "NEW|RESUME|AUTO",
  "region" : "us-east-1",
  "failOnError" : "string",
  "parallelism" : "string",
  "parserConfiguration" : {
    "baseUri" : "http://base-uri-string",
    "namedGraphUri" : "http://named-graph-string"
  },
  "updateSingleCardinalityProperties" : "string",
  "queueRequest" : "TRUE",
  "dependencies" : ["load_A_id", "load_B_id"]
}
```

Parametri della richiesta dello dello strumento di caricamento Neptune

- **`source`**: URI Amazon S3.

  Il parametro `SOURCE` accetta un URI Amazon S3 che identifica un singolo file, più file, una cartella o più cartelle. Neptune carica ogni file di dati in qualsiasi cartella specificata.

  l'URI può essere in uno dei seguenti formati.

  - `s3://bucket_name/object-key-name`
  - `https://s3.amazonaws.com/bucket_name/object-key-name`
  - `https://s3.us-east-1.amazonaws.com/bucket_name/object-key-name`

  L'`object-key-name`elemento dell'URI è equivalente al parametro prefix in una chiamata API Amazon ListObjectsS3. Identifica tutti gli oggetti nel bucket Amazon S3 specificato i cui nomi iniziano con il prefisso specificato. Può trattarsi di un singolo file o una singola cartella oppure di più file e/o cartelle.

  La cartella o le cartelle specificate possono contenere più file di vertici e più file di archi.
- **`format`**: formato dei dati. Per ulteriori informazioni sui formati di dati per il comando Neptune `Loader`, vedi Uso dello strumento di caricamento in blocco Amazon Neptune per importare i dati.

  Valori consentiti

  - **`csv`** per il formato dei dati CSV Gremlin.
  - **`opencypher`** per il formato dei dati CSV openCypher.
  - **`ntriples`** per il formato dei dati N-Triples RDF.
  - **`nquads`** per il formato dei dati N-Quads RDF.

  - **`rdfxml`** per il formato dei dati RDF\XML RDF.
  - **`turtle`** per il formato dei dati Turtle RDF.
- **`iamRoleArn`**: nome della risorsa Amazon (ARN) per un ruolo IAM che deve essere assunto dall'istanza database Neptune per l'accesso al bucket S3. Per informazioni su come creare un ruolo che abbia accesso ad Amazon S3 e su come associarlo a un cluster Neptune, vedi Prerequisiti: ruolo IAM e accesso ad Amazon S3.

  A partire dalla versione 1.2.1.0.R3 del motore, puoi anche concatenare più ruoli IAM se l'istanza DB Neptune e il bucket Amazon S3 si trovano in account diversi. AWS In questo caso, `iamRoleArn` contiene un elenco separato da virgole di ARN di ruoli, come descritto in Concatenazione di ruoli IAM in Amazon Neptune. Per esempio:

```
curl -X POST https://localhost:8182/loader \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://(the target bucket name)/(the target date file name)",
        "iamRoleArn" : "arn:aws:iam::(Account A
 ID):role/(RoleA),arn:aws:iam::(Account B ID):role/(RoleB),arn:aws:iam::(Account C
 ID):role/(RoleC)",
        "format" : "csv",
        "region" : "us-east-1"
      }'
```

- **`region`**— Il `region` parametro deve corrispondere alla regione del cluster e al bucket S3 AWS .

  Amazon Neptune è disponibile nelle seguenti regioni :

  - Stati Uniti orientali (Virginia settentrionale): `us-east-1`
  - Stati Uniti orientali (Ohio): `us-east-2`
  - Stati Uniti occidentali (California settentrionale): `us-west-1`
  - Stati Uniti occidentali (Oregon): `us-west-2`
  - Canada (Centrale): `ca-central-1`
  - Sud America (San Paolo): `sa-east-1`
  - Europa (Stoccolma): `eu-north-1`
  - Europa (Irlanda): `eu-west-1`
  - Europa (Londra): `eu-west-2`
  - Europa (Parigi): `eu-west-3`

- Europa (Francoforte): `eu-central-1`
- Medio Oriente (Bahrein): `me-south-1`
- Medio Oriente (Emirati Arabi Uniti): `me-central-1`
- Israele (Tel Aviv): `il-central-1`
- Africa (Città del Capo): `af-south-1`
- Asia Pacifico (Hong Kong): `ap-east-1`
- Asia Pacifico (Tokyo): `ap-northeast-1`
- Asia Pacifico (Seoul): `ap-northeast-2`
- Asia Pacifico (Osaka): `ap-northeast-3`
- Asia Pacifico (Singapore): `ap-southeast-1`
- Asia Pacifico (Sydney): `ap-southeast-2`
- Asia Pacifico (Mumbai): `ap-south-1`
- Cina (Pechino): `cn-north-1`
- Cina (Ningxia): `cn-northwest-1`
- AWS GovCloud (Stati Uniti occidentali): `us-gov-west-1`
- AWS GovCloud (Stati Uniti orientali): `us-gov-east-1`

- **`mode`**: modalità del processo di caricamento.

  Valori consentiti: `RESUME`, `NEW`, `AUTO`.

  Valore predefinito: `AUTO`

  - `RESUME`: in modalità RESUME, lo strumento di caricamento cerca un caricamento precedente da questa origine e, se ne trova uno, riprende l'attività di caricamento. Se non viene trovata alcuna attività di caricamento precedente, il loader si arresta.

    Il loader evita di ricaricare i file caricati correttamente in un'attività precedente. Tenta di elaborare solo i file non caricati. Se sono stati eliminati i dati caricati in precedenza dal cluster Neptune, tali dati non vengono ricaricati in questa modalità. Se un processo di caricamento precedente ha caricato correttamente tutti i file dalla stessa origine, nulla viene ricaricato e lo strumento di caricamento restituisce un risultato positivo.
  - `NEW`: in modalità NEW viene creata una nuova richiesta di caricamento, indipendentemente da eventuali caricamenti precedenti. Questa modalità può essere utilizzata per ricaricare tutti i dati

provenienti da un'origine dopo che sono stati eliminati dati caricati precedentemente dal cluster Neptune o per caricare nuovi dati disponibili nella stessa origine.

- `AUTO`: in modalità AUTO, lo strumento di caricamento cerca un'attività di caricamento precedente dalla stessa origine e, se ne trova una, riprende tale attività, proprio come in modalità `RESUME`.

  Se il loader non trova un'attività di caricamento precedente dalla stessa origine, carica tutti i dati dall'origine, proprio come in modalità `NEW`.

- **`failOnError`**: un flag per attivare un arresto completo in caso di errore.

  Valori consentiti: `"TRUE"`, `"FALSE"`.

  Valore predefinito: `"TRUE"`.

  Quando questo parametro è impostato su `"FALSE"`, il loader tenta di caricare tutti i dati nella posizione specificata, saltando eventuali voci con errori.

  Quando questo parametro è impostato su `"TRUE"`, il loader si arresta non appena rileva un errore. I dati caricati fino a quel punto persistono.

- **`parallelism`**: si tratta di un parametro facoltativo che può essere impostato per ridurre il numero di thread utilizzati dal processo di caricamento in blocco.

  Valori consentiti:

  - `LOW`: il numero di thread utilizzati è il numero di vCPU disponibili diviso per 8.
  - `MEDIUM`: il numero di thread utilizzati è il numero di vCPU disponibili diviso per 2.
  - `HIGH`: il numero di thread utilizzati corrisponde al numero di vCPU disponibili.
  - `OVERSUBSCRIBE`: il numero di thread utilizzati è il numero di vCPU disponibili moltiplicato per 2. Se viene utilizzato questo valore, il bulk loader occupa tutte le risorse disponibili.

    Ciò non significa, tuttavia, che l'impostazione `OVERSUBSCRIBE` comporti un utilizzo della CPU al 100%. Poiché l'operazione di caricamento è limitata dalle attività di I/O, l'utilizzo massimo della CPU previsto è compreso tra il 60% e il 70%.

  Valore predefinito: `HIGH`

  L'impostazione `parallelism` a volte può causare un deadlock tra i thread durante il caricamento dei dati openCypher. Quando ciò accade, Neptune restituisce l'errore `LOAD_DATA_DEADLOCK`. Di solito è possibile risolvere il problema impostando `parallelism` su un valore inferiore e riprovando il comando di caricamento.

- **`parserConfiguration`**: un oggetto opzionale con valori di configurazione del parser aggiuntivi. Ciascuno dei parametri figlio è anche facoltativo:

Nome	Valore di esempio	Descrizione
`namedGraphUri`	*`http://aws.amazon.com/neptune/vocab/v01/ DefaultNamedGraph`*	The default graph for all RDF formats when no graph is specified (for non-quads formats and NQUAD entries with no graph). The default is `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`
`baseUri`	*`http://aws.amazon.com/neptune/default`*	The base URI for RDF/XML and Turtle formats. The default is `http://aws.amazon.com/neptune/default` .

`allowEmptyStrings`	*`true`*	Gli utenti Gremlin devono essere in grado di passare valori di stringa vuoti ("") come proprietà dei nodi e degli archi durante il caricamento di dati CSV. Se `allowEmptyStrings` è impostato su `false` (valore predefinito), le stringhe vuote vengono trattate come valori null e non vengono caricate. Se `allowEmptyStrings` è impostato su `true`, lo strumento di caricamento considera le stringhe vuote come valori di proprietà validi e le carica di conseguenza.

Per ulteriori informazioni, consulta Grafo predefinito SPARQL e grafi denominati.

- **`updateSingleCardinalityProperties`**: è un parametro facoltativo che controlla il modo in cui lo strumento di caricamento in blocco tratta un nuovo valore per le proprietà di vertici o archi a cardinalità singola. Non è supportato per il caricamento di dati openCypher (vedi Caricamento di dati openCypher).

  Valori consentiti: `"TRUE"`, `"FALSE"`.

  Valore predefinito: `"FALSE"`.

  Come impostazione predefinita o quando `updateSingleCardinalityProperties` è impostato esplicitamente su `"FALSE"`, il loader considera un nuovo valore come un errore, perché viola la cardinalità singola.

  Quando `updateSingleCardinalityProperties` è impostato invece su `"TRUE"`, il bulk loader sostituisce il valore esistente con quello nuovo. Se valori di proprietà multipli edge o vertice a cardinalità singola vengono forniti nei file di origine caricati, il valore finale alla fine del caricamento

in blocco potrebbe essere uno qualsiasi di questi nuovi valori. Il loader garantisce solo che il valore esistente è stato sostituito da uno di quelli nuovi.

- **`queueRequest`**: si tratta di un parametro flag opzionale che indica se la richiesta di caricamento può essere accodata o meno.

  Non è necessario attendere il completamento di un processo di caricamento prima di emettere quello successivo, perché Neptune può accodare fino a 64 processi alla volta, a condizione che i relativi parametri `queueRequest` siano tutti impostati su `"TRUE"`. L'ordine di coda dei lavori sarà first-in-first-out (FIFO).

  Se il parametro `queueRequest` viene omesso o impostato su `"FALSE"`, la richiesta di caricamento avrà esito negativo se un'altra attività di caricamento è già in esecuzione.

  Valori consentiti: `"TRUE"`, `"FALSE"`.

  Valore predefinito: `"FALSE"`.

- **`dependencies`**: si tratta di un parametro facoltativo che può rendere subordinata una richiesta di caricamento in coda al completamento di uno o più processi precedenti nella coda.

  Neptune può accodare fino a 64 richieste di caricamento alla volta, se i relativi parametri `queueRequest` sono impostati su `"TRUE"`. Il parametro `dependencies` consente di rendere l'esecuzione di tale richiesta in coda dipendente dal completamento corretto di una o più richieste precedenti specificate nella coda.

  Ad esempio, se `Job-A` e `Job-B` del caricamento sono indipendenti l'una dall'altra, ma `Job-C` richiede che `Job-A` e `Job-B` siano completate prima del suo avvio, procedere come segue:

  1. Inviare `load-job-A` e `load-job-B` una dopo l'altra in qualsiasi ordine e salvare i loro id di caricamento.
  2. Inviare `load-job-C` con gli id di caricamento delle due attività nel campo `dependencies`:

  ```
  "dependencies" : ["job_A_load_id", "job_B_load_id"]
  ```

  A causa del parametro `dependencies`, il bulk loader non avvia `Job-C` fino a quando `Job-A` e `Job-B` non sono state completate correttamente. Se una di queste attività non riesce, l'attività non verrà eseguita e il suo stato sarà impostato su `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

È possibile impostare più livelli di dipendenza in questo modo, in modo che l'errore di un'attività causi l'annullamento di tutte le richieste direttamente o indirettamente dipendenti da essa.

- **`userProvidedEdgeIds`**: questo parametro è obbligatorio solo quando si caricano dati openCypher che contengono ID di relazione. Deve essere incluso e impostato su `True` se gli ID delle relazioni openCypher vengono forniti esplicitamente nei dati da caricare (consigliato).

  Se `userProvidedEdgeIds` è assente o è impostato su `True`, in ogni file delle relazioni all'interno del caricamento deve esistere una colonna `:ID`.

  Se `userProvidedEdgeIds` è presente ed è impostato su `False`, i file delle relazioni all'interno del caricamento non devono contenere una colonna `:ID`. Lo strumento di caricamento Neptune genera automaticamente un ID per ogni relazione.

  È utile fornire in modo esplicito gli ID delle relazioni in modo che lo strumento di caricamento possa riprendere il caricamento dopo la correzione dell'errore nei dati CSV, senza dover ricaricare le relazioni già caricate. Se gli ID delle relazioni non sono stati assegnati in modo esplicito, lo strumento di caricamento non può riprendere un caricamento non riuscito se è stato necessario correggere un file delle relazioni, perciò dovrà ricaricare tutte le relazioni.

- `accessKey`: [obsoleto] ID chiave di accesso di un ruolo IAM con accesso al bucket S3 e ai file di dati.

  Il parametro `iamRoleArn` è invece consigliato. Per informazioni su come creare un ruolo che abbia accesso ad Amazon S3 e su come associarlo a un cluster Neptune, vedi Prerequisiti: ruolo IAM e accesso ad Amazon S3.

  Per ulteriori informazioni, consulta l'argomento relativo alle chiavi di accesso (ID chiave di accesso e chiave di accesso segreta).

- `secretKey`: [obsoleto] il parametro `iamRoleArn` è invece consigliato. Per informazioni su come creare un ruolo che abbia accesso ad Amazon S3 e su come associarlo a un cluster Neptune, vedi Prerequisiti: ruolo IAM e accesso ad Amazon S3.

  Per ulteriori informazioni, consulta l'argomento relativo alle chiavi di accesso (ID chiave di accesso e chiave di accesso segreta).

Considerazioni speciali per il caricamento dei dati openCypher

- Quando si caricano dati openCypher in formato CSV, il parametro format deve essere impostato su `opencypher`.
- Il parametro `updateSingleCardinalityProperties` non è supportato per i caricamenti openCypher perché tutte le proprietà openCypher hanno cardinalità singola. Il formato di caricamento openCypher non supporta gli array e se un valore ID appare più di una volta, viene considerato un duplicato o un errore di inserimento (vedi sotto).
- Lo strumento di caricamento Neptune gestisce i duplicati che incontra nei dati openCypher come segue:
  - Se lo strumento di caricamento incontra più righe con lo stesso ID nodo, queste vengono unite utilizzando la seguente regola:
    - Tutte le etichette nelle righe vengono aggiunte al nodo.
    - Per ogni proprietà, viene caricato solo uno dei valori della proprietà. La selezione di quello da caricare non è deterministica.
  - Se lo strumento di caricamento incontra più righe con lo stesso ID relazione, ne viene caricata solo una. La selezione di quella da caricare non è deterministica.
  - Lo strumento di caricamento non aggiorna mai i valori delle proprietà di un nodo o di una relazione esistente nel database se incontra dati di caricamento con l'ID del nodo o della relazione esistente. Tuttavia, carica le etichette e le proprietà dei nodi che non sono presenti nel nodo o nella relazione esistente.
- Sebbene non sia necessario assegnare ID alle relazioni, di solito è una buona idea (vedi il parametro `userProvidedEdgeIds` sopra riportato). Senza ID relazione espliciti, lo strumento di caricamento deve ricaricare tutte le relazioni in caso di errore in un file di relazione, anziché riprendere il caricamento dal punto in cui non era riuscito.

  Inoltre, se i dati di caricamento non contengono ID relazione espliciti, lo strumento di caricamento non ha modo di rilevare relazioni duplicate.

Di seguito è illustrato un esempio di un comando di caricamento openCypher:

```
curl -X POST https://your-neptune-endpoint:port/loader \
     -H 'Content-Type: application/json' \
     -d '
     {
```

```
       "source" : "s3://bucket-name/object-key-name",
       "format" : "opencypher",
       "userProvidedEdgeIds": "TRUE",
       "iamRoleArn" : "arn:aws:iam::account-id:role/role-name",
       "region" : "region",
       "failOnError" : "FALSE",
       "parallelism" : "MEDIUM",
     }'
```

La risposta strumento di caricamento è la stessa del normale. Per esempio:

```
{
  "status" : "200 OK",
  "payload" : {
    "loadId" : "guid_as_string"
  }
}
```

Sintassi della risposta dello strumento di caricamento Neptune

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "guid_as_string"
    }
}
```

200 OK

Il processo di caricamento avviato correttamente restituisce il codice `200`.

Errori dello strumento di caricamento Neptune

Se si verifica un errore, viene restituito un oggetto JSON nel `BODY` della risposta. L'oggetto `message` contiene una descrizione dell'errore.

Categorie di errore

- `Error 400`: gli errori di sintassi restituiscono un errore di richiesta non valida HTTP `400`. Il messaggio descrive l'errore.
- `Error 500`: una richiesta valida che non può essere elaborata restituisce un errore interno del server HTTP `500`. Il messaggio descrive l'errore.

Di seguito sono riportati i possibili messaggi di errore dello strumento di caricamento con la relativa descrizione.

Messaggi di errore dello strumento di caricamento

- `Couldn't find the AWS credential for iam_role_arn` (HTTP 400)

  Non è stato possibile trovare le credenziali. Verifica le credenziali fornite sulla console o sull'output IAM. AWS CLI Assicurarsi di avere aggiunto il ruolo IAM specificato in `iamRoleArn` al cluster.
- `S3 bucket not found for source` (HTTP 400)

  Il bucket S3 non esiste. Verifica il nome del bucket.
- `The source source-uri does not exist/not reachable` (HTTP 400)

  Non sono stati trovati file corrispondenti nel bucket S3.
- `Unable to connect to S3 endpoint. Provided source = source-uri and region = aws-region` (HTTP 500)

  Non è stato possibile connettersi ad Amazon S3. La regione deve corrispondere alla regione del cluster. Verifica di disporre di un endpoint VPC. Per informazioni su come creare un endpoint VPC, vedi Creazione di un endpoint VPC Amazon S3.
- `Bucket is not in provided Region (aws-region)` (HTTP 400)

  Il bucket deve trovarsi nella stessa AWS regione dell'istanza DB di Neptune.
- `Unable to perform S3 list operation` (HTTP 400)

  L'utente o il ruolo IAM fornito non dispone di autorizzazioni `List` per il bucket o la cartella. Verifica la policy o la lista di controllo accessi (ACL) sul bucket.
- `Start new load operation not permitted on a read replica instance` (HTTP 405)

  Il caricamento è un'operazione di scrittura. Riprova a caricare sull'endpoint del cluster di lettura/scrittura.
- `Failed to start load because of unknown error from S3` (HTTP 500)

  Amazon S3 ha restituito un errore sconosciuto. Contattare AWS Support.
- `Invalid S3 access key` (HTTP 400)

  La chiave di accesso non è valida. Controlla le credenziali fornite.

- `Invalid S3 secret key` (HTTP 400)

  La chiave privata non è valida. Controlla le credenziali fornite.

- `Max concurrent load limit breached` (HTTP 400)

  Se una richiesta di caricamento viene inviata senza `"queueRequest" : "TRUE"`, e un'attività di caricamento è attualmente in esecuzione, la richiesta avrà esito negativo con questo errore.

- `Failed to start new load for the source "source name". Max load task queue size limit breached. Limit is 64` (HTTP 400)

  Neptune supporta l'accodamento di un massimo di 64 processi dello strumento di caricamento alla volta. Se una richiesta di caricamento aggiuntiva viene inviata alla coda quando contiene già 64 attività, la richiesta non riesce con questo messaggio.

Esempi di strumento di caricamento Neptune

Example Richiesta

Di seguito viene riportata una richiesta inviata tramite HTTP POST mediante il comando `curl`. La richiesta carica un file in formato CSV Neptune. Per ulteriori informazioni, consulta Formato dati di caricamento Gremlin.

```
curl -X POST \
    -H 'Content-Type: application/json' \
    https://your-neptune-endpoint:port/loader -d '
    {
      "source" : "s3://bucket-name/object-key-name",
      "format" : "csv",
      "iamRoleArn" : "ARN for the IAM role you are using",
      "region" : "region",
      "failOnError" : "FALSE",
      "parallelism" : "MEDIUM",
      "updateSingleCardinalityProperties" : "FALSE",
      "queueRequest" : "FALSE"
    }'
```

Example Risposta

```
{
    "status" : "200 OK",
```

```
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

API Get-Status dello strumento di caricamento Neptune

Consente di ottenere lo stato di un processo `loader`.

Per ottenere lo stato di caricamento, è necessario inviare una richiesta HTTP `GET` all'endpoint `https://`*`your-neptune-endpoint`*`:`*`port`*`/loader`. Per ottenere lo stato di una richiesta di caricamento specifica, è necessario includere il valore `loadId` come parametro URL oppure aggiungere `loadId` al percorso URL.

Neptune tiene traccia solo dei 1.024 processi di caricamento in blocco più recenti e memorizza solo gli ultimi 10.000 dettagli di errore per processo.

Vedi Messaggi di feed e di errore dello strumento di caricamento Neptune per un elenco dei messaggi di errore e di feed restituiti dallo strumento di caricamento in caso di errori.

Indice

- Esempio di un errore Data prefetch task interrupted

Richieste Get-Status dello strumento di caricamento Neptune

Sintassi della richiesta Get-Status dello strumento di caricamento

```
GET https://your-neptune-endpoint:port/loader?loadId=loadId
```

```
GET https://your-neptune-endpoint:port/loader/loadId
```

```
GET https://your-neptune-endpoint:port/loader
```

Parametri della richiesta Get-Status dello strumento di caricamento Neptune

- **`loadId`**: ID del processo di caricamento. Se non si specifica un `loadId`, viene restituito un elenco di ID di caricamento.
- **`details`**: include i dettagli aggiuntivi rispetto allo stato generale.

  Valori consentiti: `TRUE`, `FALSE`.

  Valore predefinito: `FALSE`.
- **`errors`**: include l'elenco degli errori.

  Valori consentiti: `TRUE`, `FALSE`.

  Valore predefinito: `FALSE`.

  L'elenco degli errori è paginato. I parametri `page` ed `errorsPerPage` consentono di esaminare tutti gli errori.
- **`page`**: numero della pagina dell'errore. È valido solo con il parametro `errors` impostato su `TRUE`.

  Valori consentiti: interi positivi

  Valore predefinito: 1
- **`errorsPerPage`**: numero di errori per ogni pagina. È valido solo con il parametro `errors` impostato su `TRUE`.

  Valori consentiti: interi positivi

Valore predefinito: 10

- **`limit`**: numero di ID di caricamento da elencare. È valido solo quando si richiede un elenco di ID di caricamento inviando una richiesta `GET` senza specificare alcun `loadId`.

  Valori consentiti: interi positivi da 1 a 100.

  Valore predefinito: 100

- **`includeQueuedLoads`**: parametro facoltativo che può essere utilizzato per escludere gli ID di caricamento delle richieste di caricamento in coda quando viene richiesto un elenco di ID di caricamento.

  > **Note**
  >
  > Questo parametro è disponibile a partire dal rilascio 1.0.3.0 del motore Neptune.

  Per impostazione predefinita, gli ID di caricamento di tutte le attività di caricamento con stato `LOAD_IN_QUEUE` sono inclusi in tale elenco. Vengono visualizzati prima degli ID di caricamento di altre attività, ordinati in base al momento in cui le attività sono state aggiunte alla coda dalla più recente alla più vecchia.

  Valori consentiti: `TRUE`, `FALSE`.

  Valore predefinito: `TRUE`.

Risposte Get-Status dello strumento di caricamento Neptune

Layout JSON delle risposte Get-Status dello strumento di caricamento Neptune

Il layout generale di una risposta dello stato dello strumento di caricamento è il seguente:

```
{
    "status" : "200 OK",
    "payload" : {
        "feedCount" : [
            {
                "LOAD_FAILED" : number
            }
        ],
```

```
        "overallStatus" : {
            "fullUri" : "s3://bucket/key",
            "runNumber" : number,
            "retryNumber" : number,
            "status" : "string",
            "totalTimeSpent" : number,
            "startTime" : number,
            "totalRecords" : number,
            "totalDuplicates" : number,
            "parsingErrors" : number,
            "datatypeMismatchErrors" : number,
            "insertErrors" : number,
        },
        "failedFeeds" : [
            {
                "fullUri" : "s3://bucket/key",
                "runNumber" : number,
                "retryNumber" : number,
                "status" : "string",
                "totalTimeSpent" : number,
                "startTime" : number,
                "totalRecords" : number,
                "totalDuplicates" : number,
                "parsingErrors" : number,
                "datatypeMismatchErrors" : number,
                "insertErrors" : number,
            }
        ],
        "errors" : {
            "startIndex" : number,
            "endIndex" : number,
            "loadId" : "string,
            "errorLogs" : [ ]
        }
    }
}
```

Oggetti di risposta Get-Status **`overallStatus`** e **`failedFeeds`** dello strumento di caricamento Neptune

Le possibili risposte restituite per ogni feed non riuscito, incluse le descrizioni degli errori, sono le stesse dell'oggetto `overallStatus` in una risposta `Get-Status`.

I seguenti campi vengono visualizzati nell'oggetto `overallStatus` per tutti i caricamenti e nell'oggetto `failedFeeds` per ogni feed non riuscito.

- **`fullUri`**: URI del file o dei file da caricare.

  Tipo: stringa

  Formato: `s3://`*`bucket`*`/`*`key`*.

- **`runNumber`**: numero di esecuzione di questo caricamento o feed. Aumenta quando il carico viene riavviato.

  Tipo: unsigned long

- **`retryNumber`**: numero di nuovi tentativi di questo caricamento o feed. Aumenta quando lo strumento di caricamento prova nuovamente a eseguire un feed o un caricamento automaticamente.

  Tipo: unsigned long

- **`status`**: stato restituito del caricamento o del feed. `LOAD_COMPLETED` indica un caricamento riuscito senza problemi. Per un elenco di altri messaggi sullo stato del caricamento, vedi Messaggi di feed e di errore dello strumento di caricamento Neptune.

  Tipo: stringa.

- **`totalTimeSpent`**: tempo, in secondi, impiegato per analizzare e inserire i dati per il caricamento o il feed. Non è incluso il tempo impiegato per recuperare l'elenco dei file di origine.

  Tipo: unsigned long

- **`totalRecords`**: totale dei record caricati o che si è provato a caricare.

  Tipo: unsigned long

  Tenere presente che quando si carica da un file CSV, il conteggio dei record non si riferisce al numero di righe caricate, ma piuttosto al numero di singoli record in quelle righe. Ad esempio, prendiamo un piccolo file CSV come questo:

```
~id,~label,name,team
'P-1','Player','Stokes','England'
```

  Neptune considererà questo file come contenente 3 record, ovvero:

```
P-1  label Player
P-1  name  Stokes
P-1  team  England
```

- **`totalDuplicates`**: numero di record duplicati rilevati.

  Tipo: unsigned long

  Come nel caso del conteggio `totalRecords`, questo valore contiene il numero di singoli record duplicati in un file CSV, non il numero di righe duplicate. Prendiamo ad esempio questo piccolo file CSV:

```
~id,~label,name,team
P-2,Player,Kohli,India
P-2,Player,Kohli,India
```

  Lo stato restituito dopo il caricamento sarà simile al seguente, con un totale di 6 record, di cui 3 duplicati:

```
{
  "status": "200 OK",
  "payload": {
    "feedCount": [
      {
        "LOAD_COMPLETED": 1
      }
    ],
    "overallStatus": {
      "fullUri": "(the URI of the CSV file)",
      "runNumber": 1,
      "retryNumber": 0,
      "status": "LOAD_COMPLETED",
      "totalTimeSpent": 3,
      "startTime": 1662131463,
      "totalRecords": 6,
      "totalDuplicates": 3,
      "parsingErrors": 0,
      "datatypeMismatchErrors": 0,
      "insertErrors": 0
    }
  }
```

```
}
```

Per i caricamenti openCypher, un duplicato viene conteggiato quando:

- Lo strumento di caricamento rileva che una riga in un file dei nodi ha un ID senza uno spazio ID uguale a un altro valore ID senza uno spazio ID, in un'altra riga o appartenente a un nodo esistente.
- Lo strumento di caricamento rileva che una riga in un file dei nodi ha un ID con uno spazio ID uguale a un altro valore ID con uno spazio ID, in un'altra riga o appartenente a un nodo esistente.

Per informazioni, consulta Considerazioni speciali per il caricamento dei dati openCypher.

- **`parsingErrors`**: numero di errori di analisi rilevati.

  Tipo: unsigned long
- **`datatypeMismatchErrors`**: numero di record il cui tipo di dati non corrisponde ai dati specificati.

  Tipo: unsigned long
- **`insertErrors`**: numero di record che non è stato possibile inserire a causa di errori.

  Tipo: unsigned long

Oggetto di risposta Get-Status **`errors`** dello strumento di caricamento Neptune

Gli errori rientrano nelle seguenti categorie:

- **`Error 400`**: un oggetto `loadId` non valido restituisce un errore di richiesta non valida HTTP `400`. Il messaggio descrive l'errore.
- **`Error 500`**: una richiesta valida che non può essere elaborata restituisce un errore interno del server HTTP `500`. Il messaggio descrive l'errore.

Vedi Messaggi di feed e di errore dello strumento di caricamento Neptune per un elenco dei messaggi di errore e di feed restituiti dallo strumento di caricamento in caso di errori.

Se si verifica un errore, viene restituito un oggetto JSON `errors` nell'oggetto `BODY` della risposta con i seguenti campi:

- **`startIndex`**: indice del primo errore incluso.

Tipo: unsigned long

- **`endIndex`**: indice dell'ultimo errore incluso.

  Tipo: unsigned long

- **`loadId`**: ID del caricamento. È possibile usare questo ID per stampare gli errori per il caricamento impostando il parametro `errors` su `TRUE`.

  Tipo: stringa.

- **`errorLogs`**: elenco degli errori.

  Type: elenco

Oggetto di risposta Get-Status **`errorLogs`** dello strumento di caricamento Neptune

L'oggetto `errorLogs` contenuto nell'oggetto `errors` della risposta Get-Status dello strumento di caricamento contiene un oggetto che descrive ogni errore utilizzando i seguenti campi:

- **`errorCode`**: identifica la natura dell'errore.

  Può accettare uno dei seguenti valori:

  - `PARSING_ERROR`
  - `S3_ACCESS_DENIED_ERROR`
  - `FROM_OR_TO_VERTEX_ARE_MISSING`
  - `ID_ASSIGNED_TO_MULTIPLE_EDGES`
  - `SINGLE_CARDINALITY_VIOLATION`
  - `FILE_MODIFICATION_OR_DELETION_ERROR`
  - `OUT_OF_MEMORY_ERROR`
  - `INTERNAL_ERROR` (restituito quando lo strumento di caricamento in blocco non è in grado di determinare il tipo di errore).

- **`errorMessage`**: messaggio che descrive l'errore.

  Può trattarsi di un messaggio generico associato al codice di errore o di un messaggio specifico contenente dettagli, ad esempio su un vertice da/a mancante o su un errore di analisi.

- **`fileName`**: nome del feed.

- **`recordNum`**: in caso di errore di analisi, questo è il numero di record nel file del record che non è stato possibile analizzare. Viene impostato su zero se il numero di record non è applicabile all'errore o se non è stato possibile determinarlo.

Ad esempio, lo strumento di caricamento in blocco genererà un errore di analisi se rileva una riga in errore come la seguente in un file RDF `nquads`:

```
<http://base#subject> |http://base#predicate> <http://base#true> .
```

Come si può notare, il secondo `http` nella riga precedente dovrebbe essere preceduto da < anziché da `|`. L'oggetto di errore risultante in `errorLogs` in una risposta di stato sarà simile al seguente:

```
{
    "errorCode" : "PARSING_ERROR",
    "errorMessage" : "Expected '<', found: |",
    "fileName" : "s3://bucket/key",
    "recordNum" : 12345
},
```

Esempi di Get-Status dello strumento di caricamento Neptune

Esempio di richiesta dello stato di caricamento

Di seguito viene riportata una richiesta inviata tramite HTTP `GET` mediante il comando `curl`.

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)'
```

Example Risposta

```
{
    "status" : "200 OK",
    "payload" : {
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
```

```
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        }
    }
}
```

Esempio di richiesta di loadIds

Di seguito viene riportata una richiesta inviata tramite HTTP `GET` mediante il comando `curl`.

```
curl -X GET 'https://your-neptune-endpoint:port/loader?limit=3'
```

Example Risposta

```
{
    "status" : "200 OK",
    "payload" : {
         "loadIds" : [
            "a2c0ce44-a44b-4517-8cd4-1dc144a8e5b5",
            "09683a01-6f37-4774-bb1b-5620d87f1931",
            "58085eb8-ceb4-4029-a3dc-3840969826b9"
        ]
    }
}
```

Esempio di richiesta dello stato dettagliato

Di seguito viene riportata una richiesta inviata tramite HTTP `GET` mediante il comando `curl`.

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)?details=true'
```

Example Risposta

```
{
    "status" : "200 OK",
```

```
    "payload" : {
        "failedFeeds" : [
            {
                "datatypeMismatchErrors" : 0,
                "fullUri" : "s3://bucket/key",
                "insertErrors" : 0,
                "parsingErrors" : 5,
                "retryNumber" : 0,
                "runNumber" : 1,
                "status" : "LOAD_FAILED",
                "totalDuplicates" : 0,
                "totalRecords" : 5,
                "totalTimeSpent" : 3.0
            }
        ],
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        }
    }
}
```

Esempi di Get-Status **`errorLogs`** dello strumento di caricamento Neptune

Esempio di risposta di stato dettagliata in caso di errori

Questa è una richiesta inviata tramite HTTP `GET` utilizzando `curl`:

```
curl -X GET 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-c29ce5f54802?details=true&errors=true&page=1&errorsPerPage=3'
```

Example di una risposta dettagliata in caso di errori

Questo è un esempio della risposta che potresti ottenere dalla query precedente, con un oggetto `errorLogs` che elenca gli errori di caricamento riscontrati:

```
{
    "status" : "200 OK",
    "payload" : {
        "failedFeeds" : [
            {
                "datatypeMismatchErrors" : 0,
                "fullUri" : "s3://bucket/key",
                "insertErrors" : 0,
                "parsingErrors" : 5,
                "retryNumber" : 0,
                "runNumber" : 1,
                "status" : "LOAD_FAILED",
                "totalDuplicates" : 0,
                "totalRecords" : 5,
                "totalTimeSpent" : 3.0
            }
        ],
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        },
        "errors" : {
            "endIndex" : 3,
            "errorLogs" : [
                {
                    "errorCode" : "PARSING_ERROR",
```

```
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 1
                },
                {
                    "errorCode" : "PARSING_ERROR",
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 2
                },
                {
                    "errorCode" : "PARSING_ERROR",
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 3
                }
            ],
            "loadId" : "0a237328-afd5-4574-a0bc-c29ce5f548002",
            "startIndex" : 1
        }
    }
}
```

Esempio di un errore **`Data prefetch task interrupted`**

Occasionalmente quando ottieni uno stato `LOAD_FAILED` e richiedi informazioni più dettagliate, l'errore restituito potrebbe essere un `PARSING_ERROR` con un messaggio `Data prefetch task interrupted`, come questo:

```
"errorLogs" : [
    {
        "errorCode" : "PARSING_ERROR",
        "errorMessage" : "Data prefetch task interrupted: Data prefetch task for 11467
 failed",
        "fileName" : "s3://some-source-bucket/some-source-file",
        "recordNum" : 0
    }
]
```

Questo errore ha luogo quando si è verificata un interruzione temporaneo nel processo di caricamento dei dati che non è stato in genere causata dalla richiesta o dai dati. Di solito può essere risolto semplicemente eseguendo nuovamente la richiesta di caricamento in blocco. Se stai

utilizzando impostazioni predefinite, ovvero `"mode":"AUTO"` e `"failOnError":"TRUE"`, il loader salta i file che ha già caricato e riprende il caricamento dei file che non aveva ancora caricato quando si è verificata l'interruzione.

Annullamento di un processo dello strumento di caricamento Neptune

Annulla un processo di caricamento.

Per annullare un lavoro, è necessario inviare una richiesta HTTP `DELETE` all'endpoint `https://`*`your-neptune-endpoint`*`:`*`port`*`/loader`. Il valore `loadId` può essere aggiunto al percorso URL `/loader` o incluso nell'URL come variabile.

Sintassi della richiesta di annullamento di un processo

```
DELETE https://your-neptune-endpoint:port/loader?loadId=loadId
```

```
DELETE https://your-neptune-endpoint:port/loader/loadId
```

Parametri della richiesta del processo di annullamento

loadId

L'ID del processo di caricamento.

Sintassi della risposta del processo di annullamento

```
no response body
```

200 OK

Il processo di caricamento eliminato correttamente restituisce il codice `200`.

Errori del processo di annullamento

Se si verifica un errore, viene restituito un oggetto JSON nel `BODY` della risposta. L'oggetto `message` contiene una descrizione dell'errore.

Categorie di errore

- **`Error 400`**: un oggetto `loadId` non valido restituisce un errore di richiesta non valida HTTP `400`. Il messaggio descrive l'errore.

- **`Error 500`**: una richiesta valida che non può essere elaborata restituisce un errore interno del server HTTP `500`. Il messaggio descrive l'errore.

Messaggi di errore del processo di annullamento

Di seguito sono riportati i possibili messaggi di errore dell'API di annullamento con la relativa descrizione.

- `The load with id = load_id does not exist or not active` (HTTP 404): il caricamento non è stato trovato. Verifica il valore del parametro `id`.
- `Load cancellation is not permitted on a read replica instance.` (HTTP 405): il caricamento è un'operazione di scrittura. Riprova a caricare sull'endpoint del cluster di lettura/scrittura.

Esempi del processo di annullamento

Example Richiesta

Di seguito viene riportata una richiesta inviata tramite HTTP `DELETE` mediante il comando `curl`.

```
curl -X DELETE 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-c29ce5f54802'
```

Utilizzo AWS Database Migration Service per caricare dati in Amazon Neptune da un altro data store

AWS Database Migration Service (AWS DMS) può caricare dati in Neptune dai database di origine supportati in modo rapido e sicuro. Il database di origine resterà completamente operativo anche durante la migrazione, per ridurre al minimo le interruzioni delle applicazioni che lo utilizzano.

Informazioni dettagliate AWS DMS in merito sono disponibili nella Guida per l'AWS Database Migration Service utente e nell'API Reference.AWS Database Migration Service In particolare, è possibile trovare informazioni su come impostare un cluster Neptune come destinazione per la migrazione in Utilizzo di Amazon Neptune come destinazione per AWS Database Migration Service.

Di seguito sono riportati alcuni prerequisiti per l'importazione di dati in Neptune usando AWS DMS:

- Dovrai creare un oggetto di mappatura delle AWS DMS tabelle per definire come estrarre i dati dal database di origine (vedi Specificare la selezione e le trasformazioni delle tabelle mediante la mappatura delle tabelle utilizzando JSON nella Userguide per i dettagli). AWS DMS Questo oggetto di configurazione della mappatura della tabella specifica quali tabelle devono essere lette e in quale ordine e come vengono denominate le colonne. Può anche filtrare le righe copiate e fornire semplici trasformazioni di valore come la conversione in minuscolo o l'arrotondamento.
- Sarà necessario creare un oggetto Neptune `GraphMappingConfig` per specificare come devono essere caricati i dati estratti dal database di origine in Neptune. Per i dati RDF (interrogati utilizzando SPARQL), `GraphMappingConfig` viene scritto nel linguaggio di mappatura R2RML standard di W3. Per i dati dei grafi di proprietà (sottoposti a query mediante Gremlin), `GraphMappingConfig` è un oggetto JSON, descritto in GraphMappingConfig Layout per dati Property-Graph/Gremlin.
- È necessario utilizzare AWS DMS per creare un'istanza di replica nello stesso VPC del cluster Neptune DB, per mediare il trasferimento dei dati.
- Sarà inoltre necessario utilizzare un bucket Amazon S3 come spazio di archiviazione intermedio per la gestione temporanea dei dati di migrazione.

Creare un Nettuno GraphMappingConfig

L'oggetto `GraphMappingConfig` creato specifica come i dati estratti da un datastore di origine devono essere caricati in un cluster database Neptune. Il suo formato varia a seconda che sia destinato al caricamento di dati RDF o al caricamento di dati del grafo delle proprietà.

Per i dati RDF, è possibile utilizzare il linguaggio W3 R2RML per mappare i dati relazionali a RDF.

Se si stanno caricando i dati del grafico delle proprietà per essere interrogati utilizzando Gremlin, si crea un oggetto JSON per `GraphMappingConfig`.

GraphMappingConfig Layout per dati RDF/SPARQL

Se si stanno caricando dati RDF per essere interrogati utilizzando SPARQL, si scrive `GraphMappingConfig` in R2RML. `R2RML` è un linguaggio W3 standard per la mappatura dei dati relazionali a RDF. Ecco un esempio:

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/ns#> .

<#TriplesMap1>
```

```
    rr:logicalTable [ rr:tableName "nodes" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{id}";
        rr:class ex:Employee;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "label" ];
    ] .
```

Ecco un altro esempio:

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<#TriplesMap2>
    rr:logicalTable [ rr:tableName "Student" ];
    rr:subjectMap   [ rr:template "http://example.com/{ID}{Name}";
                      rr:class foaf:Person ];
    rr:predicateObjectMap [
        rr:predicate ex:id ;
        rr:objectMap  [ rr:column "ID";
                        rr:datatype xsd:integer ]
    ];
    rr:predicateObjectMap [
        rr:predicate foaf:name ;
        rr:objectMap  [ rr:column "Name" ]
    ] .
```

La raccomandazione W3 a R2RML: RDB to RDF Mapping Language fornisce i dettagli della lingua.

GraphMappingConfig Layout per dati Property-Graph/Gremlin

Un oggetto comparabile `GraphMappingConfig` per i dati del grafico delle proprietà è un oggetto JSON che fornisce una regola di mapping per ogni entità grafico da generare dai dati di origine. Il modello seguente mostra l'aspetto di ogni regola in questo oggetto:

```
{
  "rules": [
    {
      "rule_id": "(an identifier for this rule)",
```

```
    "rule_name": "(a name for this rule)",
    "table_name": "(the name of the table or view being loaded)",
    "vertex_definitions": [
      {
        "vertex_id_template": "{col1}",
        "vertex_label": "(the vertex to create)",
        "vertex_definition_id": "(an identifier for this vertex)",
        "vertex_properties": [
          {
            "property_name": "(name of the property)",
            "property_value_template": "{col2} or text",
            "property_value_type": "(data type of the property)"
          }
        ]
      }
    ]
  },
  {
    "rule_id": "(an identifier for this rule)",
    "rule_name": "(a name for this rule)",
    "table_name": "(the name of the table or view being loaded)",
    "edge_definitions": [
      {
        "from_vertex": {
          "vertex_id_template": "{col1}",
          "vertex_definition_id": "(an identifier for the vertex referenced above)"
        },
        "to_vertex": {
          "vertex_id_template": "{col3}",
          "vertex_definition_id": "(an identifier for the vertex referenced above)"
        },
        "edge_id_template": {
          "label": "(the edge label to add)",
          "template": "{col1}_{col3}"
        },
        "edge_properties":[
          {
            "property_name": "(the property to add)",
            "property_value_template": "{col4} or text",
            "property_value_type": "(data type like String, int, double)"
          }
        ]
      }
    ]
```

```
    }
  ]
}
```

Si noti che la presenza di un'etichetta di vertice implica che il vertice viene creato qui, mentre la sua assenza implica che il vertice viene creato da una sorgente diversa e questa definizione aggiunge solo proprietà di vertice.

Ecco una regola di esempio per un record dipendente:

```
{
  "rules": [
    {
      "rule_id": "1",
      "rule_name": "vertex_mapping_rule_from_nodes",
      "table_name": "nodes",
      "vertex_definitions": [
        {
          "vertex_id_template": "{emp_id}",
          "vertex_label": "employee",
          "vertex_definition_id": "1",
          "vertex_properties": [
            {
              "property_name": "name",
              "property_value_template": "{emp_name}",
              "property_value_type": "String"
            }
          ]
        }
      ]
    },
    {
      "rule_id": "2",
      "rule_name": "edge_mapping_rule_from_emp",
      "table_name": "nodes",
      "edge_definitions": [
        {
          "from_vertex": {
            "vertex_id_template": "{emp_id}",
            "vertex_definition_id": "1"
          },
          "to_vertex": {
            "vertex_id_template": "{mgr_id}",
```

```
            "vertex_definition_id": "1"
          },
          "edge_id_template": {
            "label": "reportsTo",
            "template": "{emp_id}_{mgr_id}"
          },
          "edge_properties":[
            {
              "property_name": "team",
              "property_value_template": "{team}",
              "property_value_type": "String"
            }
          ]
        }
      ]
    }
  ]
}
```

Creazione di un'attività AWS DMS di replica con Neptune come obiettivo

Dopo aver creato le configurazioni di mapping delle tabelle e mapping dei grafi, utilizzare il seguente processo per caricare i dati dall'archivio di origine in Neptune. Consulta la AWS DMS documentazione per maggiori dettagli sulle API in questione.

Fase 1: Creare un'istanza di AWS DMS replica

Crea un'istanza di AWS DMS replica nel VPC su cui è in esecuzione il cluster Neptune DB (vedi Lavorare con AWS un'istanza di replica DMS e nella Guida per l'utente). CreateReplicationInstance AWS DMS A tale scopo, è possibile utilizzare un AWS CLI comando come il seguente:

```
aws dms create-replication-instance \
    --replication-instance-identifier (the replication instance identifier) \
    --replication-instance-class (the size and capacity of the instance, like
 'dms.t2.medium') \
    --allocated-storage (the number of gigabytes to allocate for the instance
 initially) \
    --engine-version (the DMS engine version that the instance should use) \
    --vpc-security-group-ids (the security group to be used with the instance)
```

Fase 2: Crea un AWS DMS endpoint per il database di origine

Il passaggio successivo consiste nel creare un AWS DMS endpoint per il data store di origine. Puoi utilizzare l' AWS DMS CreateEndpointAPI in questo AWS CLI modo:

```
aws dms create-endpoint \
    --endpoint-identifier (source endpoint identifier) \
    --endpoint-type source \
    --engine-name (name of source database engine) \
    --username (user name for database login) \
    --password (password for login) \
    --server-name (name of the server) \
    --port (port number) \
    --database-name (database name)
```

Fase 3. Impostare un bucket Amazon S3 per Neptune da utilizzare per i dati di gestione temporanea

Se non si dispone di un bucket Amazon S3 che è possibile utilizzare per i dati di gestione temporanea, crearne uno come spiegato in Creazione di un bucket nella Guida introduttiva di Amazon S3 o in Come creare un bucket S3? nella Guida per l'utente della console.

Sarà necessario creare una policy IAM che conceda le autorizzazioni `GetObject`, `PutObject`, `DeleteObject`e `ListObject` al bucket se non ne esiste già una:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::(bucket-name)"
      ]
    },
    {
      "Sid": "AllObjectActions",
      "Effect": "Allow",
```

```
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:ListObject"
       ],
      "Resource": [
        "arn:aws:s3:::(bucket-name)/*"
      ]
    }
  ]
}
```

Se l'autenticazione IAM del cluster database Neptune è abilitata, sarà necessario includere anche le policy seguenti:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "(the ARN of your Neptune DB cluster resource)"
    }
  ]
}
```

Creare un ruolo IAM come documento di attendibilità a cui collegare la policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "dms.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Sid": "neptune",
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Dopo aver collegato la policy al ruolo, collegare il ruolo al cluster database Neptune. Ciò consentirà di AWS DMS utilizzare il bucket per l'archiviazione temporanea dei dati in fase di caricamento.

Fase 4. Creare un endpoint Amazon S3 nel VPC Neptune

Creare ora un endpoint Gateway VPC per il bucket Amazon S3 intermedio nel VPC in cui si trova il cluster Neptune. È possibile utilizzare AWS Management Console o the AWS CLI per eseguire questa operazione, come descritto in Creazione di un endpoint gateway.

Fase 5. Crea un endpoint AWS DMS di destinazione per Neptune

Crea un AWS DMS endpoint per il tuo cluster Neptune DB di destinazione. Puoi usare l' AWS DMS CreateEndpointAPI con il `NeptuneSettings` parametro in questo modo:

```
aws dms create-endpoint \
    --endpoint-identifier (target endpoint identifier) \
    --endpoint-type target \
    --engine-name neptune \
    --server-name (name of the server) \
    --port (port number) \
    --neptune-settings '{ \
      "ServiceAccessRoleArn": "(ARN of the service access role)", \
      "S3BucketName": "(name of S3 bucket to use for staging files when migrating)", \
      "S3BucketFolder": "(name of the folder to use in that S3 bucket)", \
      "ErrorRetryDuration": (number of milliseconds to wait between bulk-load retries),
 \
      "MaxRetryCount": (the maximum number of times to retry a failing bulk-load job),
 \
      "MaxFileSize": (maximum file size, in bytes, of the staging files written to S3),
 \
      "isIamAuthEnabled": (set to true if IAM authentication is enabled on the Neptune
 cluster) }'
```

L'oggetto JSON passato all' AWS DMS `CreateEndpoint`API nel relativo `NeptuneSettings` parametro ha i seguenti campi:

- **`ServiceAccessRoleArn`**: (obbligatorio) ARN di un ruolo IAM che consente l'accesso con granularità fine al bucket S3 utilizzato per la migrazione dei dati in Neptune. Questo ruolo dovrebbe inoltre disporre delle autorizzazioni per accedere al cluster database Neptune se è abilitata l'autorizzazione IAM.
- **`S3BucketName`**: (obbligatorio) per la migrazione con caricamento completo, l'istanza di replica converte tutti i dati RDS in file CSV di quadruple e li carica in questo bucket di gestione temporanea in S3 e quindi li carica in blocco in Neptune.
- **`S3BucketFolder`**: (obbligatorio) cartella da utilizzare nel bucket di gestione temporanea S3.
- **`ErrorRetryDuration`**: (facoltativo) numero di millisecondi di attesa dopo che una richiesta Neptune ha esito negativo prima di effettuare una richiesta di nuovo tentativo. Il valore di default è 250.
- **`MaxRetryCount`**— (opzionale) Il numero massimo di richieste di nuovi tentativi da AWS DMS effettuare dopo un errore riprovevole. Il predefinito è 5.
- **`MaxFileSize`**: (facoltativo) dimensione massima in byte di ogni file di gestione temporanea salvato in S3 durante la migrazione. Il valore predefinito è 1.048.576 KB (1 GB).
- **`IsIAMAuthEnabled`**: (facoltativo) impostare su `true` se l'autenticazione IAM è abilitata sul cluster database Neptune o su `false` in caso contrario. Il valore predefinito è `false`.

Fase 6. Connessioni di test ai nuovi endpoint

Puoi testare la connessione a ciascuno di questi nuovi endpoint utilizzando l' AWS DMS TestConnectionAPI in questo modo:

```
aws dms test-connection \
    --replication-instance-arn (the ARN of the replication instance) \
    --endpoint-arn (the ARN of the endpoint you are testing)
```

Fase 7. Crea un'attività di AWS DMS replica

Dopo aver completato con successo i passaggi precedenti, crea un'attività di replica per la migrazione dei dati dal tuo data store di origine a Neptune, utilizzando l'API in questo modo: AWS DMS CreateReplicationTask

```
aws dms create-replication-task \
    --replication-task-identifier (name for the replication task) \
    --source-endpoint-arn (ARN of the source endpoint) \
    --target-endpoint-arn (ARN of the target endpoint) \
    --replication-instance-arn (ARN of the replication instance) \
    --migration-type full-load \
    --table-mappings (table-mapping JSON object or URI like 'file:///tmp/table-
mappings,json') \
    --task-data (a GraphMappingConfig object or URI like 'file:///tmp/graph-mapping-
config.json')
```

Il parametro `TaskData` fornisce l'oggetto GraphMappingConfig che specifica come i dati copiati devono essere archiviati in Neptune.

Fase 8. Avviate l'attività di replica AWS DMS

Ora è possibile avviare l'attività di replica:

```
aws dms start-replication-task
    --replication-task-arn (ARN of the replication task started in the previous step)
    --start-replication-task-type start-replication
```

Esecuzione di query su un grafo Neptune

Neptune supporta i seguenti linguaggi di query a grafo per accedere a un grafo:

- Gremlin, definito da Apache TinkerPop per la creazione e l'interrogazione di grafici delle proprietà.

  Una query in Gremlin è un attraversamento composto da passaggi discreti, ognuno dei quali segue un arco fino a un nodo.

  Vedi Accesso al grafo Neptune con Gremlin per ottenere informazioni sull'uso di Gremlin in Neptune e Conformità agli standard Gremlin in Amazon Neptune per trovare dettagli specifici sull'implementazione di Gremlin in Neptune.
- openCypher è un linguaggio di query dichiarativo per grafi di proprietà che è stato sviluppato originariamente da Neo4j, per poi diventare open source nel 2015, e che ha contribuito al progetto openCypher con una licenza open source Apache 2. La sintassi è documentata nella specifica openCypher.
- SPARQL è un linguaggio dichiarativo basato sulla corrispondenza di modelli del grafo, per l'esecuzione di query sui dati RDF. È supportato dal World Wide Web Consortium.

  Vedi Accesso al grafo Neptune con SPARQL per ottenere informazioni sull'uso di SPARQL in Neptune e Conformità agli standard SPARQL in Amazon Neptune per trovare dettagli specifici sull'implementazione di SPARQL in Neptune.

> **Note**
>
> Sia Gremlin che openCypher possono essere usati per eseguire query sui dati dei grafi di proprietà archiviati in Neptune, indipendentemente da come sono stati caricati.

Argomenti

Accodamento delle query in Amazon Neptune

Durante lo sviluppo e l'ottimizzazione di applicazioni grafiche, può essere utile conoscere le implicazioni di come le query vengono accodate dal database. In Amazon Neptune, l'accodamento delle query si verifica come segue:

- Il numero massimo di query che è possibile accodare per istanza, indipendentemente dalla dimensione dell'istanza, è 8.192. Tutte le query superiori a quel numero vengono rifiutate e danno esito negativo con un `ThrottlingException`.
- Il numero massimo di query che è possibile eseguire contemporaneamente è determinato dal numero di thread di lavoro assegnati, che in genere è impostato su due volte il numero di core CPU virtuali disponibili.
- La latenza delle query include il tempo trascorso da una query nella coda, nonché il round trip della rete e il tempo necessario per l'esecuzione.

Determinare quante query ci sono nella coda in un dato momento

La `MainRequestQueuePendingRequests` CloudWatch metrica registra il numero di richieste in attesa nella coda di input con una granularità di cinque minuti (vedi). Metriche di Neptune CloudWatch

Per Gremlin, è possibile ottenere un conteggio corrente di query nella coda utilizzando il valore `acceptedQueryCount` restituito da API di stato delle query Gremlin. Si noti, tuttavia, che il valore `acceptedQueryCount` restituito da API di stato delle query SPARQL include tutte le query accettate dall'avvio del server, incluse le query completate.

Come l'accodamento delle query può influire sui timeout

Come notato in precedenza, la latenza delle query include il tempo impiegato da una query nella coda e il tempo necessario per l'esecuzione.

Poiché il periodo di timeout di una query viene generalmente misurato a partire da quando entra nella coda, una coda a spostamento lento può mandare molte query in timeout non appena vengono rimosse dalla coda. Questo è ovviamente indesiderabile, quindi è bene evitare di accodare un gran numero di query a meno che non possano essere eseguite rapidamente.

Accesso al grafo Neptune con Gremlin

Amazon Neptune è compatibile con TinkerPop Apache 3 e Gremlin. Ciò significa che è possibile connettersi a un'istanza DB di Neptune e utilizzare il linguaggio di attraversamento Gremlin per interrogare il grafico (vedere The Graph nella documentazione di Apache 3). TinkerPop Per le differenze nell'implementazione di Gremlin in Neptune, consulta Conformità agli standard Gremlin.

Versioni diverse del motore Neptune supportano versioni diverse di Gremlin. Controlla la pagina di rilascio del motore della versione di Neptune che stai utilizzando per determinare quale rilascio di Gremlin supporta.

L'attraversamento in Gremlin corrisponde a una serie di passaggi concatenati. Inizia in un vertice (o edge). Percorre il grafo seguendo gli edge in uscita di ogni vertice, quindi gli edge in uscita di quei vertici. Nell'attraversamento, ogni passaggio corrisponde a un'operazione. Per ulteriori informazioni, consulta la documentazione di The Traversal in the 3. TinkerPop

Per l'accesso a Gremlin esistono varianti di linguaggio e supporto in diversi linguaggi di programmazione. Per ulteriori informazioni, consulta On Gremlin Language Variants nella documentazione 3. TinkerPop

Questa documentazione descrive come accedere a Neptune con le varianti e i linguaggi di programmazione seguenti.

Come illustrato in Crittografia in transito: connessione a Neptune utilizzando il protocollo SSL/HTTPS, è necessario utilizzare TLS/SSL (Transport Layer Security/Secure Sockets Layer) quando ci si connette a Neptune in tutte le regioni AWS .

Gremlin-Groovy

Gli esempi di Console Gremlin e HTTP REST in questa sezione utilizzano la variante Gremlin-Groovy. Per ulteriori informazioni sulla console Gremlin e su Amazon Neptune, consulta la sezione the section called “Utilizzo di Gremlin” del Quick Start.

Gremlin-Java

L'esempio Java è stato scritto con l'implementazione ufficiale di Java TinkerPop 3 e utilizza la variante Gremlin-Java.

Gremlin-Python

L'esempio Python è scritto con l'implementazione ufficiale di TinkerPop Python 3 e utilizza la variante Gremlin-Python.

Le sezioni seguenti descrivono come utilizzare la console Gremlin, REST su HTTPS e vari linguaggi di programmazione per connettersi a un'istanza database Neptune.

Prima di iniziare, devi disporre di quanto segue:

- Istanza database Neptune. Per informazioni sulla creazione di un'istanza database Neptune, consulta Creazione di un nuovo cluster database Neptune.
- Istanza Amazon EC2 nello stesso cloud privato virtuale (VPC) dell'istanza database Neptune.

Per ulteriori informazioni sul caricamento di dati in Neptune, incluso i prerequisiti, i formati di caricamento e i parametri di caricamento, vedi Caricamento di dati in Amazon Neptune.

Argomenti

Configurazione della console Gremlin per la connessione a un'istanza database Neptune

La Gremlin Console consente di sperimentare TinkerPop grafici e interrogazioni in un ambiente REPL (loop). read-eval-print

Installazione della console Gremlin e connessione ad essa nel modo consueto

Con la console Gremlin puoi connetterti a un database a grafo remoto. La sezione seguente illustra l'installazione e la configurazione della console Gremlin per connettersi in remoto a un'istanza database Neptune. Segui queste istruzioni da un'istanza Amazon EC2 nello stesso cloud privato virtuale (VPC) dell'istanza database Neptune.

Per informazioni sulla connessione a Neptune con SSL/TLS (obbligatorio), consulta Configurazione SSL/TLS.

> **Note**
>
> Se nel cluster database Neptune è abilitata l'autenticazione IAM, per installare la console Gremlin seguire le istruzioni riportate in Connessione a Neptune tramite la console Gremlin con la firma di Signature Version 4 anziché le istruzioni riportate qui.

Per installare la console Gremlin e connettersi a Neptune

1. I binari della console Gremlin richiedono Java 8 o Java 11. Queste istruzioni presuppongono l'uso di Java 11. Puoi installare Java 11 sull'istanza EC2 come segue:

   - Se utilizzi Amazon Linux 2 (AL2):

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

   - Se utilizzi Amazon Linux 2023 (AL2023):

     ```
     sudo yum install java-11-amazon-corretto-devel
     ```

   - Per altre distribuzioni, usa l'opzione appropriata tra le seguenti:

     ```
     sudo yum install java-11-openjdk-devel
     ```

oppure:

```
sudo apt-get install openjdk-11-jdk
```

2. Per impostare Java 11 come runtime predefinito sull'istanza EC2, digita quanto segue.

```
sudo /usr/sbin/alternatives --config java
```

Quando richiesto, immetti il numero per Java 11.

3. Scarica la versione appropriata della console Gremlin dal sito Web di Apache. Controlla la pagina di rilascio del motore della versione del motore Neptune che stai utilizzando per determinare quale versione di Gremlin supporta. Ad esempio, per la versione 3.6.5, puoi scaricare la console Gremlin dal sito Web di Apache Tinkerpop3 sull'istanza EC2 in questo modo:

```
wget https://archive.apache.org/dist/tinkerpop/3.6.5/apache-tinkerpop-gremlin-
console-3.6.5-bin.zip
```

4. Decomprimi il file zip della console Gremlin.

```
unzip apache-tinkerpop-gremlin-console-3.6.5-bin.zip
```

5. Cambiare directory nella directory decompressa.

```
cd apache-tinkerpop-gremlin-console-3.6.5
```

6. Nella sottodirectory `conf` della directory estratta, creare un file denominato `neptune-remote.yaml` con il testo seguente. Sostituisci *`your-neptune-endpoint`* con il nome host o l'indirizzo IP dell'istanza DB di Neptune. Le parentesi quadre (`[ ]`) sono obbligatorie.

> **Note**
>
> Per informazioni su come trovare il nome host dell'istanza database Neptune, consulta la sezione Connessione agli endpoint Amazon Neptune.

```
hosts: [your-neptune-endpoint]
port: 8182
```

```
connectionPool: { enableSsl: true }
serializer: { className:
 org.apache.tinkerpop.gremlin.driver.ser.GraphBinaryMessageSerializerV1,
              config: { serializeResultToString: true }}
```

7. In un terminale, vai alla directory della console Gremlin (`apache-tinkerpop-gremlin-console-3.6.5`) e immetti il comando seguente per eseguire la console Gremlin.

```
bin/gremlin.sh
```

   Verrà visualizzato l'output seguente:

```
         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin>
```

   Ora ti trovi al prompt `gremlin>`. Immetti i restanti passaggi in questo prompt.

8. Al prompt `gremlin>`, immetti quanto segue per connetterti all'istanza database Neptune.

```
:remote connect tinkerpop.server conf/neptune-remote.yaml
```

9. Al prompt `gremlin>`, immetti quanto segue per passare alla modalità remota. In questo modo tutte le query Gremlin vengono inviate alla connessione remota.

```
:remote console
```

10. Immetti quanto segue per inviare una query al grafo Gremlin.

```
g.V().limit(1)
```

11. Al termine, immetti quanto segue per uscire dalla console di Gremlin.

```
:exit
```

> **Note**
>
> Utilizzare un punto e virgola (`;`) o un carattere nuova riga (`\n`) per separare ogni istruzione. Ogni attraversamento che precede l'attraversamento finale deve terminare in `next()` per essere eseguito. Vengono restituiti solo i dati dell'ultimo attraversamento.

Per ulteriori informazioni sull'implementazione di Gremlin in Neptune, consulta the section called “Conformità agli standard Gremlin”.

Un modo alternativo per connettersi alla console Gremlin

Svantaggi dell'approccio di connessione normale

Il modo più comune per connettersi alla console Gremlin è quello spiegato sopra, utilizzando comandi come questo al prompt `gremlin>`.

```
gremlin> :remote connect tinkerpop.server conf/(file name).yaml
gremlin> :remote console
```

Funziona bene e consente di inviare query a Neptune. Tuttavia, esclude il motore di script Groovy dal ciclo, quindi Neptune tratta tutte le query come puro Gremlin. Ciò significa che i seguenti moduli di query hanno esito negativo:

```
gremlin> 1 + 1
gremlin> x = g.V().count()
```

La cosa più vicina all'utilizzo di una variabile quando si è connessi in questo modo è utilizzare la variabile `result` gestita dalla console e inviare la query utilizzando `:>`, in questo modo:

```
gremlin> :remote console
==>All scripts will now be evaluated locally - type ':remote console' to return
 to remote mode for Gremlin Server - [krl-1-cluster.cluster-ro-cm9t6tfwbtsr.us-
east-1.neptune.amazonaws.com/172.31.19.217:8182]
gremlin> :> g.V().count()
==>4249

gremlin> println(result)
[result{object=4249 class=java.lang.Long}]
```

```
gremlin> println(result['object'])
[4249]
```

Un modo diverso di connettersi

Puoi anche connetterti alla console Gremlin in un modo diverso, che potresti trovare più comodo, come questo:

```
gremlin> g = traversal().withRemote('conf/neptune.properties')
```

In questo caso il formato di `neptune.properties` è il seguente:

```
gremlin.remote.remoteConnectionClass=org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteCon
gremlin.remote.driver.clusterFile=conf/my-cluster.yaml
gremlin.remote.driver.sourceName=g
```

Il file `my-cluster.yaml` sarà simile al seguente:

```
hosts: [my-cluster-abcdefghijk.us-east-1.neptune.amazonaws.com]
port: 8182
serializer: { className:
 org.apache.tinkerpop.gremlin.driver.ser.GraphBinaryMessageSerializerV1,
              config: { serializeResultToString: false } }
connectionPool: { enableSsl: true }
```

La configurazione della connessione alla console Gremlin in questo modo consente di effettuare correttamente i seguenti tipi di query:

```
gremlin> 1+1
==>2

gremlin> x=g.V().count().next()
==>4249

gremlin> println("The answer was ${x}")
The answer was 4249
```

Puoi evitare di visualizzare il risultato, in questo modo:

```
gremlin> x=g.V().count().next();[]
gremlin> println(x)
4249
```

Tutti i metodi usuali di esecuzione di query (senza il passaggio terminale) continuano a funzionare. Per esempio:

```
gremlin> g.V().count()
==>4249
```

Puoi anche usare il passaggio `g.io().read()` per caricare un file con questo tipo di connessione.

Utilizzo dell'endpoint HTTPS REST per connettersi a un'istanza database Neptune

Amazon Neptune fornisce un endpoint HTTPS per le query Gremlin. L'interfaccia REST è compatibile con qualsiasi versione di Gremlin utilizzata dal cluster database (consulta la pagina di rilascio del motore della versione del motore Neptune che stai utilizzando per determinare quale rilascio di Gremlin supporta).

> **Note**
>
> Come illustrato in Crittografia in transito: connessione a Neptune utilizzando il protocollo SSL/HTTPS, Neptune ora richiede la connessione tramite HTTPS anziché HTTP.

Le istruzioni seguenti mostrano come connettersi a un endpoint di Gremlin utilizzando il comando `curl` e HTTPS. Segui queste istruzioni da un'istanza Amazon EC2 nello stesso cloud privato virtuale (VPC) dell'istanza database Neptune.

L'endpoint HTTPS per le query Gremlin in un'istanza database Neptune è `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin`.

> **Note**
>
> Per informazioni su come trovare il nome host dell'istanza database Neptune, consulta Connessione agli endpoint Amazon Neptune.

Per connettersi a Neptune utilizzando l'endpoint HTTP REST

L'esempio seguente utilizza curl per inviare una query Gremlin tramite HTTP POST. La query viene inviata in formato JSON nel corpo del post come proprietà `gremlin`.

```
curl -X POST -d '{"gremlin":"g.V().limit(1)"}' https://your-neptune-endpoint:port/
gremlin
```

Questo esempio restituisce il primo vertice nel grafo utilizzando l'attraversamento `g.V().limit(1)`. Per eseguire una query su qualcos'altro, sostituirlo con un altro attraversamento Gremlin.

> **Important**
>
> Per impostazione predefinita, l'endpoint REST restituisce tutti i risultati in un unico insieme di risultati JSON. Se questo insieme di risultati è troppo grande, può verificarsi un'eccezione `OutOfMemoryError` sull'istanza database Neptune.
> È possibile evitare tale problema abilitando le risposte in blocchi (risultati restituiti in una serie di risposte separate). Per informazioni, consulta Utilizzo di intestazioni HTTP finali opzionali per abilitare le risposte Gremlin in più parti.

Sebbene le richieste HTTP POST siano consigliate per l'invio di query Gremlin, è possibile utilizzare anche le richieste HTTP GET:

```
curl -G "https://your-neptune-endpoint:port?gremlin=g.V().count()"
```

> **Note**
>
> Neptune non supporta la proprietà `bindings`.

Utilizzo di intestazioni HTTP finali opzionali per abilitare le risposte Gremlin in più parti

Per impostazione predefinita, la risposta HTTP alle query Gremlin viene restituita in un unico insieme di risultati JSON. Nel caso di un insieme di risultati molto grande, ciò può causare un'eccezione `OutOfMemoryError` sull'istanza database.

Tuttavia, è possibile abilitare le risposte in blocchi (risposte restituite in più parti separate). A tale scopo, includere nella richiesta un'intestazione transfer-encoding (TE) trailers (`te: trailers`). Per ulteriori informazioni sulle intestazioni TE, vedi la pagina MDN sulle intestazioni di richiesta TE.

Quando una risposta viene restituita in più parti, può essere difficile diagnosticare un problema che si verifica dopo la ricezione della prima parte, poiché la prima parte arriva con il codice di stato HTTP `200` (OK). Un errore successivo di solito si traduce in un corpo del messaggio contenente una risposta danneggiata, al termine della quale Neptune aggiunge un messaggio di errore.

Per facilitare il rilevamento e la diagnosi di questo tipo di errore, Neptune include anche due nuovi campi di intestazione all'interno delle intestazioni finali di ogni blocco di risposta:

- `X-Neptune-Status`: contiene il codice di risposta seguito da un nome breve. Ad esempio, in caso di esito positivo, l'intestazione finale sarà: `X-Neptune-Status: 200 OK`. In caso di errore, il codice di risposta sarà uno dei codici di errore del motore Neptune, ad esempio `X-Neptune-Status: 500 TimeLimitExceededException`.
- `X-Neptune-Detail`: è vuoto per le richieste riuscite. In caso di errori, contiene il messaggio di errore JSON. Poiché nei valori di intestazione HTTP sono consentiti solo caratteri ASCII, la stringa JSON è codificata come URL.

> **Note**
>
> Neptune attualmente non supporta la compressione `gzip` delle risposte in blocchi. Se il client richiede contemporaneamente la codifica e la compressione in blocchi, Neptune salta la compressione.

Client Gremlin basati su Java da utilizzare con Amazon Neptune

Puoi utilizzare uno dei due client Gremlin open source basati su Java con Amazon Neptune: il client Apache TinkerPop Java Gremlin o il client Gremlin per Amazon Neptune.

Client Apache Java Gremlin TinkerPop

Se puoi, usa sempre l'ultima versione del client Apache TinkerPop Java Gremlin supportata dalla versione del tuo motore. Le versioni più recenti contengono numerose correzioni di bug che migliorano la stabilità, le prestazioni e l'usabilità del client.

La tabella seguente elenca le prime e le ultime versioni del TinkerPop client supportate da diverse versioni del motore Neptune:

Versione del motore Neptune	Versione minima TinkerPop	TinkerPop Versione massima
`1.3.1.0`	`3.6.2`	`3.6.5`
`1.3.0.0`	`3.6.2`	`3.6.4`
`1.2.1.1`	`3.6.2`	`3.6.2`
`1.2.1.0`	`3.6.2`	`3.6.2`
`1.2.0.2`	`3.5.2`	`3.5.6`
`1.2.0.1`	`3.5.2`	`3.5.6`
`1.2.0.0`	`3.5.2`	`3.5.6`
`1.1.1.0`	`3.5.2`	`3.5.6`
`1.1.0.0`	`3.4.0`	`3.4.13`
`1.0.5.1` e versioni precedenti	(obsoleta)	(obsoleta)

TinkerPop i client sono in genere retrocompatibili all'interno di una serie (`3.3.x`, ad esempio, o`3.4.x`). Esistono casi eccezionali in cui è necessario interrompere la compatibilità con le versioni precedenti, quindi è meglio controllare i consigli di aggiornamento prima di TinkerPop eseguire l'aggiornamento a una nuova versione del client.

Il client potrebbe non essere in grado di utilizzare i nuovi passaggi o le nuove funzionalità introdotti nelle versioni successive a quella supportata dal server, ma è possibile aspettarsi che le query e le funzionalità esistenti funzionino a meno che la raccomandazione di aggiornamento non richieda una modifica sostanziale.

> **Note**
>
> A partire dalla versione 1.1.1.0 del motore Neptune, non utilizzare una versione precedente a. TinkerPop `3.5.2`

Gli utenti di Python dovrebbero evitare di usare TinkerPop version a `3.4.9` causa di un'impostazione di timeout predefinita che richiede una configurazione diretta (vedi TINKERPOP-2505).

Client Java Gremlin per Amazon Neptune

Il client Gremlin per Amazon Neptune è un client Gremlin open source basato su Java che funge da sostituto diretto del client Java standard. TinkerPop

Il client Gremlin di Neptune è ottimizzato per i cluster Neptune. Consente di gestire la distribuzione del traffico su più istanze di un cluster e si adatta alle modifiche della topologia del cluster quando si aggiunge o si rimuove una replica. È anche possibile configurare il client per distribuire le richieste su un sottoinsieme di istanze del cluster, in base al ruolo, al tipo di istanza, alla zona di disponibilità (AZ) o ai tag associati alle istanze.

L'ultima versione del client Java Gremlin di Neptune è disponibile su Maven Central.

Per ulteriori informazioni sul client Java Gremlin di Neptune, consulta questo post del blog. Per esempi di codice e demo, consulta il progetto del cliente. GitHub

Utilizzo di un client Java per connettersi a un'istanza database Neptune

La sezione seguente illustra l'esecuzione di un esempio Java completo che si connette a un'istanza DB di Neptune ed esegue un attraversamento di Gremlin utilizzando il client Apache Gremlin. TinkerPop

Segui queste istruzioni da un'istanza Amazon EC2 nello stesso cloud privato virtuale (VPC) dell'istanza database Neptune.

Per connettersi a Neptune tramite Java

1. Installare Apache Maven sull'istanza EC2. Per aggiungere un archivio con un pacchetto Maven, immettere prima quanto segue:

   ```
   sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Per impostare il numero di versione per i pacchetti, immettere quanto segue.

```
sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
```

Potrai quindi utilizzare yum per installare Maven:

```
sudo yum install -y apache-maven
```

2. Installare Java. Le librerie Gremlin richiedono Java 8 o 11. Puoi installare Java 11 nel modo seguente:

   - Se utilizzi Amazon Linux 2 (AL2):

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

   - Se utilizzi Amazon Linux 2023 (AL2023):

     ```
     sudo yum install java-11-amazon-corretto-devel
     ```

   - Per altre distribuzioni, usa l'opzione appropriata tra le seguenti:

     ```
     sudo yum install java-11-openjdk-devel
     ```

     oppure:

     ```
     sudo apt-get install openjdk-11-jdk
     ```

3. Imposta Java 11 come runtime predefinito sull'istanza EC2: inserisci quanto segue per impostare Java 8 come runtime predefinito sull'stanza EC2:

   ```
   sudo /usr/sbin/alternatives --config java
   ```

   Quando richiesto, immetti il numero per Java 11.

4. Crea una nuova directory denominata **`gremlinjava`**:

   ```
   mkdir gremlinjava
   cd gremlinjava
   ```

5. Nella directory `gremlinjava`, creare un file `pom.xml`, quindi aprirlo in un editor di testo:

```
nano pom.xml
```

6. Copiare quanto segue nel file `pom.xml` e salvare:

```
<project xmlns="https://maven.apache.org/POM/4.0.0"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://
maven.apache.org/maven-v4_0_0.xsd">
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.amazonaws</groupId>
  <artifactId>GremlinExample</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>GremlinExample</name>
  <url>https://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>org.apache.tinkerpop</groupId>
      <artifactId>gremlin-driver</artifactId>
      <version>3.6.5</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.apache.tinkerpop/gremlin-groovy
      (Not needed for TinkerPop version 3.5.2 and up)
    <dependency>
      <groupId>org.apache.tinkerpop</groupId>
      <artifactId>gremlin-groovy</artifactId>
      <version>3.6.5</version>
    </dependency> -->
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-jdk14</artifactId>
      <version>1.7.25</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
```

```
        <version>2.5.1</version>
        <configuration>
          <source>11</source>
          <target>11</target>
        </configuration>
      </plugin>
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>exec-maven-plugin</artifactId>
          <version>1.3</version>
          <configuration>
            <executable>java</executable>
            <arguments>
              <argument>-classpath</argument>
              <classpath/>
              <argument>com.amazonaws.App</argument>
            </arguments>
            <mainClass>com.amazonaws.App</mainClass>
            <complianceLevel>1.11</complianceLevel>
            <killAfter>-1</killAfter>
          </configuration>
        </plugin>
    </plugins>
  </build>
</project>
```

> **Note**
>
> Se stai modificando un progetto Maven esistente, la dipendenza necessaria è evidenziata nel codice precedente.

7. Creare sottodirectory per il codice sorgente di esempio (`src/main/java/com/amazonaws/`) digitando quanto segue nella riga di comando:

```
mkdir -p src/main/java/com/amazonaws/
```

8. Nella directory `src/main/java/com/amazonaws/`, creare un file denominato `App.java`, quindi aprirlo in un editor di testo.

```
nano src/main/java/com/amazonaws/App.java
```

9. Copiare quanto segue nel file `App.java`. Sostituisci *`your-neptune-endpoint`* con l'indirizzo della tua istanza DB Neptune. Non includere il prefisso `https://` nel metodo `addContactPoint`.

> **Note**
>
> Per informazioni su come trovare il nome host dell'istanza database Neptune, consulta Connessione agli endpoint Amazon Neptune.

```
package com.amazonaws;
import org.apache.tinkerpop.gremlin.driver.Cluster;
import org.apache.tinkerpop.gremlin.driver.Client;
import
 org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversalSource;
import org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal;
import static
 org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource.traversal;
import org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteConnection;
import org.apache.tinkerpop.gremlin.structure.T;

public class App
{
  public static void main( String[] args )
  {
    Cluster.Builder builder = Cluster.build();
    builder.addContactPoint("your-neptune-endpoint");
    builder.port(8182);
    builder.enableSsl(true);

    Cluster cluster = builder.create();

    GraphTraversalSource g =
 traversal().withRemote(DriverRemoteConnection.using(cluster));

    // Add a vertex.
    // Note that a Gremlin terminal step, e.g. iterate(), is required to make a
 request to the remote server.
    // The full list of Gremlin terminal steps is at https://tinkerpop.apache.org/
docs/current/reference/#terminal-steps
    g.addV("Person").property("Name", "Justin").iterate();
```

```
    // Add a vertex with a user-supplied ID.
    g.addV("Custom Label").property(T.id, "CustomId1").property("name", "Custom id
 vertex 1").iterate();
    g.addV("Custom Label").property(T.id, "CustomId2").property("name", "Custom id
 vertex 2").iterate();

    g.addE("Edge Label").from(__.V("CustomId1")).to(__.V("CustomId2")).iterate();

    // This gets the vertices, only.
    GraphTraversal t = g.V().limit(3).elementMap();

    t.forEachRemaining(
      e ->  System.out.println(t.toList())
    );

    cluster.close();
  }
}
```

Per informazioni sulla connessione a Neptune con SSL/TLS (obbligatorio), consulta Configurazione SSL/TLS.

10. Compilare ed eseguire il campione usando il comando Maven seguente:

```
mvn compile exec:exec
```

L'esempio precedente restituisce una mappa della chiave e i valori di ogni proprietà per i primi due vertici nel grafo utilizzando l'attraversamento `g.V().limit(3).elementMap()`. Per eseguire query per qualcos'altro, sostituirla con un altro attraversamento Gremlin con uno dei metodi finali appropriati.

> **Note**
>
> La parte finale della query Gremlin, `.toList()` è obbligatoria per inviare l'attraversamento al server per la valutazione. Se non includi quel metodo o un altro metodo equivalente, la query non viene inviata all'istanza database Neptune.
> È inoltre necessario aggiungere una chiusura appropriata quando si aggiunge un vertice o un arco, ad esempio quando si utilizza il passaggio `addV( )`.

I metodi riportati sotto inviano la query all'istanza database Neptune:

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

Configurazione SSL/TLS per il client Java Gremlin

Neptune richiede che SSL/TLS sia abilitato per impostazione predefinita. In genere, se il driver Java è configurato con `enableSsl(true)`, può connettersi a Neptune senza dover impostare `trustStore()` o `keyStore()` con una copia locale di un certificato. Nelle versioni precedenti era TinkerPop consigliato l'uso di `keyCertChainFile()` per configurare un `.pem` file archiviato localmente, ma questo è diventato obsoleto e non è più disponibile dopo la 3.5.x. Se stavi usando tale configurazione con un certificato pubblico, usando `SFSRootCAG2.pem`, ora puoi rimuovere la copia locale.

Tuttavia, se l'istanza con cui ti stai connettendo non dispone di una connessione Internet tramite la quale verificare un certificato pubblico o se il certificato che stai utilizzando non è pubblico, puoi effettuare le seguenti operazioni per configurare una copia locale del certificato:

Configurazione di una copia locale del certificato per abilitare SSL/TLS

1. Scarica e installa keytool da Oracle. Ciò renderà molto più semplice la configurazione dell'archivio chiavi locale.
2. Scarica il certificato CA `SFSRootCAG2.pem` (l'SDK Gremlin Java richiede un certificato per verificare il certificato remoto):

   ```
   wget https://www.amazontrust.com/repository/SFSRootCAG2.pem
   ```

3. Crea un archivio chiavi in formato JKS o PKCS12. Questo esempio usa JKS. Rispondi alle domande che seguono al prompt. La password che crei qui sarà necessaria in seguito:

   ```
   keytool -genkey -alias (host name) -keyalg RSA -keystore server.jks
   ```

4. Importa il file `SFSRootCAG2.pem` che hai scaricato nell'archivio chiavi appena creato:

```
keytool -import -keystore server.jks -file .pem
```

5. Configura l'oggetto `Cluster` a livello di codice:

```
Cluster cluster = Cluster.build("(your neptune endpoint)")
                         .port(8182)
                         .enableSSL(true)
                         .keyStore('server.jks')
                         .keyStorePassword("(the password from step 2)")
                         .create();
```

Si può fare la stessa cosa in un file di configurazione, come è possibile fare con la console Gremlin:

```
hosts: [(your neptune endpoint)]
port: 8182
connectionPool: { enableSsl: true, keyStore: server.jks, keyStorePassword: (the
 password from step 2) }
serializer: { className:
 org.apache.tinkerpop.gremlin.driver.ser.GraphBinaryMessageSerializerV1, config:
 { serializeResultToString: true }}
```

Esempio Java di connessione a un'istanza database Neptune con logica di riconnessione

Il seguente esempio Java mostra come connettersi al client Gremlin con la logica di riconnessione per eseguire il ripristino in seguito a una disconnessione imprevista.

Presenta le seguenti dipendenze:

```
<dependency>
    <groupId>org.apache.tinkerpop</groupId>
    <artifactId>gremlin-driver</artifactId>
    <version>${gremlin.version}</version>
</dependency>

<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>amazon-neptune-sigv4-signer</artifactId>
```

```
    <version>${sig4.signer.version}</version>
</dependency>

<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>amazon-neptune-gremlin-java-sigv4</artifactId>
    <version>${sig4.signer.version}</version>
</dependency>

<dependency>
    <groupId>com.evanlennick</groupId>
    <artifactId>retry4j</artifactId>
    <version>0.15.0</version>
</dependency>
```

Ecco il codice di esempio:

```
public static void main(String args[]) {
  boolean useIam = true;

  // Create Gremlin cluster and traversal source
  Cluster.Builder builder = Cluster.build()
         .addContactPoint(System.getenv("neptuneEndpoint"))
         .port(Integer.parseInt(System.getenv("neptunePort")))
         .enableSsl(true)
         .minConnectionPoolSize(1)
         .maxConnectionPoolSize(1)
         .serializer(Serializers.GRAPHBINARY_V1D0)
         .reconnectInterval(2000);

  if (useIam) {
    builder = builder.channelizer(SigV4WebSocketChannelizer.class);
  }

  Cluster cluster = builder.create();

  GraphTraversalSource g = AnonymousTraversalSource
      .traversal()
      .withRemote(DriverRemoteConnection.using(cluster));

  // Configure retries
  RetryConfig retryConfig = new RetryConfigBuilder()
      .retryOnCustomExceptionLogic(getRetryLogic())
```

```
      .withDelayBetweenTries(1000, ChronoUnit.MILLIS)
      .withMaxNumberOfTries(5)
      .withFixedBackoff()
      .build();

  @SuppressWarnings("unchecked")
  CallExecutor<Object> retryExecutor = new CallExecutorBuilder<Object>()
      .config(retryConfig)
      .build();

  // Do lots of queries
  for (int i = 0; i < 100; i++){
    String id = String.valueOf(i);

    @SuppressWarnings("unchecked")
    Callable<Object> query = () -> g.V(id)
        .fold()
        .coalesce(
            unfold(),
            addV("Person").property(T.id, id))
        .id().next();

    // Retry query
    // If there are connection failures, the Java Gremlin client will automatically
    // attempt to reconnect in the background, so all we have to do is wait and retry.
    Status<Object> status = retryExecutor.execute(query);

    System.out.println(status.getResult().toString());
  }

  cluster.close();
}

private static Function<Exception, Boolean> getRetryLogic() {

  return e -> {

    Class<? extends Exception> exceptionClass = e.getClass();

    StringWriter stringWriter = new StringWriter();
    String message = stringWriter.toString();


    if (RemoteConnectionException.class.isAssignableFrom(exceptionClass)){
```

```
      System.out.println("Retrying because RemoteConnectionException");
      return true;
    }

    // Check for connection issues
    if (message.contains("Timed out while waiting for an available host") ||
        message.contains("Timed-out waiting for connection on Host") ||
        message.contains("Connection to server is no longer active") ||
        message.contains("Connection reset by peer") ||
        message.contains("SSLEngine closed already") ||
        message.contains("Pool is shutdown") ||
        message.contains("ExtendedClosedChannelException") ||
        message.contains("Broken pipe") ||
        message.contains(System.getenv("neptuneEndpoint")))
    {
      System.out.println("Retrying because connection issue");
      return true;
    };

    // Concurrent writes can sometimes trigger a ConcurrentModificationException.
    // In these circumstances you may want to backoff and retry.
    if (message.contains("ConcurrentModificationException")) {
      System.out.println("Retrying because ConcurrentModificationException");
      return true;
    }

    // If the primary fails over to a new instance, existing connections to the old
 primary will
    // throw a ReadOnlyViolationException. You may want to back and retry.
    if (message.contains("ReadOnlyViolationException")) {
      System.out.println("Retrying because ReadOnlyViolationException");
      return true;
    }

    System.out.println("Not a retriable error");
    return false;
  };
}
```

Utilizzo di Python per connettersi a un'istanza database Neptune

Se puoi, usa sempre l'ultima versione del client Apache TinkerPop Python Gremlin, gremlinpython, supportata dalla versione del tuo motore. Le nuove versioni contengono numerose correzioni di bug

che migliorano la stabilità, le prestazioni e la fruibilità del client. La `gremlinpython` versione da utilizzare in genere si allinea alle TinkerPop versioni descritte nella tabella per il client Java Gremlin.

> **Note**
>
> Le versioni `gremlinpython` 3.5.x sono compatibili con le versioni TinkerPop 3.4.x purché si utilizzino solo le funzionalità 3.4.x nelle query Gremlin che si scrivono.

La sezione seguente illustra come eseguire un esempio Python che si connette a un'istanza database Amazon Neptune ed esegue un attraversamento Gremlin.

Segui queste istruzioni da un'istanza Amazon EC2 nello stesso cloud privato virtuale (VPC) dell'istanza database Neptune.

Prima di iniziare, esegui queste attività:

- Scarica e installa Python 3.6 o versione successiva dal sito Web Python.org.
- Verificare di aver installato pip. Se non si dispone di pip o non si è sicuri, consultare Do I need to install pip? (Devo installare pip?) nella documentazione pip.
- Se l'installazione Python non lo comprende già, scaricare `futures` come segue: `pip install futures`

Per connettersi a Neptune tramite Python

1. Installare il pacchetto `gremlinpython` immettendo quanto segue:

   ```
   pip install --user gremlinpython
   ```

2. Creare un file denominato `gremlinexample.py`, quindi aprirlo in un editor di testo.
3. Copiare quanto segue nel file `gremlinexample.py`. Sostituisci *`your-neptune-endpoint`* con l'indirizzo della tua istanza DB Neptune.

   Per informazioni su come trovare l'indirizzo dell'istanza database Neptune, consulta la sezione Connessione agli endpoint Amazon Neptune.

   ```
   from __future__  import print_function  # Python 2/3 compatibility
   ```

```
from gremlin_python import statics
from gremlin_python.structure.graph import Graph
from gremlin_python.process.graph_traversal import __
from gremlin_python.process.strategies import *
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection

graph = Graph()

remoteConn = DriverRemoteConnection('wss://your-neptune-endpoint:8182/gremlin','g')
g = graph.traversal().withRemote(remoteConn)

print(g.V().limit(2).toList())
remoteConn.close()
```

4. Immettere il comando seguente per eseguire l'esempio:

```
python gremlinexample.py
```

La query Gremlin alla fine di questo esempio restituisce i vertici (`g.V().limit(2)`) in un elenco. Questo elenco viene quindi stampato con la funzione Python standard `print`.

> **Note**
>
> La parte finale della query Gremlin, `toList()` è obbligatoria per inviare l'attraversamento al server per la valutazione. Se non includi quel metodo o un altro metodo equivalente, la query non viene inviata all'istanza database Neptune.

I metodi riportati sotto inviano la query all'istanza database Neptune:

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

L'esempio precedente restituisce i primi due vertici del grafo utilizzando l'attraversamento `g.V().limit(2).toList()`. Per eseguire query per qualcos'altro, sostituirla con un altro attraversamento Gremlin con uno dei metodi finali appropriati.

Utilizzo di .NET per connettersi a un'istanza database Neptune

Se puoi, usa sempre l'ultima versione del client TinkerPop Apache.NET Gremlin, Gremlin.Net, supportata dalla versione del tuo motore. Le nuove versioni contengono numerose correzioni di bug che migliorano la stabilità, le prestazioni e la fruibilità del client. La `Gremlin.Net` versione da utilizzare in genere si allinea alle TinkerPop versioni descritte nella tabella per il client Java Gremlin.

La sezione seguente contiene un codice di esempio scritto in C # che si connette a un'istanza database Neptune ed esegue un attraversamento Gremlin.

Le connessioni ad Amazon Neptune devono provenire da un'istanza Amazon EC2 che si trova nello stesso cloud privato virtuale (VPC) dell'istanza database Neptune. Questo codice di esempio è stato testato su un'istanza Amazon EC2 che esegue Ubuntu.

Prima di iniziare, esegui queste attività:

- Installa .NET sull'istanza Amazon EC2. Per istruzioni su come installare .NET su sistemi operativi multipli, incluso Windows, Linux e macOS, vedere Get Started with .NET.
- Installa Gremlin.NET eseguendo `dotnet add package gremlin.net` per il tuo pacchetto. Per ulteriori informazioni, consulta Gremlin.net nella documentazione. TinkerPop

Per connettersi a Neptune utilizzando Gremlin.NET

1. Crea un nuovo progetto .NET.

```
dotnet new console -o gremlinExample
```

2. Spostare le directory nella nuova directory di progetto.

```
cd gremlinExample
```

3. Copiare quanto segue nel file `Program.cs`. Sostituisci *`your-neptune-endpoint`*con l'indirizzo della tua istanza DB Neptune.

Per informazioni su come trovare l'indirizzo dell'istanza database Neptune, consulta la sezione Connessione agli endpoint Amazon Neptune.

```
using System;
using System.Threading.Tasks;
using System.Collections.Generic;
using Gremlin.Net;
using Gremlin.Net.Driver;
using Gremlin.Net.Driver.Remote;
using Gremlin.Net.Structure;
using static Gremlin.Net.Process.Traversal.AnonymousTraversalSource;
namespace gremlinExample
{
  class Program
  {
    static void Main(string[] args)
    {
      try
      {
        var endpoint = "your-neptune-endpoint";
        // This uses the default Neptune and Gremlin port, 8182
        var gremlinServer = new GremlinServer(endpoint, 8182, enableSsl: true );
        var gremlinClient = new GremlinClient(gremlinServer);
        var remoteConnection = new DriverRemoteConnection(gremlinClient, "g");
        var g = Traversal().WithRemote(remoteConnection);
        g.AddV("Person").Property("Name", "Justin").Iterate();
        g.AddV("Custom Label").Property("name", "Custom id vertex 1").Iterate();
        g.AddV("Custom Label").Property("name", "Custom id vertex 2").Iterate();
        var output = g.V().Limit<Vertex>(3).ToList();
        foreach(var item in output) {
            Console.WriteLine(item);
        }
      }
      catch (Exception e)
      {
          Console.WriteLine("{0}", e);
      }
    }
  }
}
```

4. Immettere il comando seguente per eseguire l'esempio:

```
dotnet run
```

La query Gremlin alla fine di questo esempio restituisce il conteggio di un singolo vertice a scopo di test. Viene quindi stampata sulla console.

> **Note**
>
> La parte finale della query Gremlin, `Next()` è obbligatoria per inviare l'attraversamento al server per la valutazione. Se non includi quel metodo o un altro metodo equivalente, la query non viene inviata all'istanza database Neptune.

I metodi riportati sotto inviano la query all'istanza database Neptune:

- `ToList()`
- `ToSet()`
- `Next()`
- `NextTraverser()`
- `Iterate()`

Utilizza `Next()` se hai bisogno che i risultati della query vengano serializzati e restituiti oppure `Iterate()` in caso contrario.

L'esempio precedente restituisce un elenco utilizzando l'attraversamento `g.V().Limit(3).ToList()`. Per eseguire query per qualcos'altro, sostituirla con un altro attraversamento Gremlin con uno dei metodi finali appropriati.

Utilizzo di Node.js per connettersi a un'istanza database Neptune

Se puoi, usa sempre la versione più recente del client Apache TinkerPop JavaScript Gremlin, gremlin, supportata dalla versione del tuo motore. Le nuove versioni contengono numerose correzioni di bug che migliorano la stabilità, le prestazioni e la fruibilità del client. La versione di `gremlin` da usare in genere si allinea alle TinkerPop versioni descritte nella tabella per il client Java Gremlin.

La sezione seguente illustra come eseguire un esempio Node.js che si connette a un'istanza database Amazon Neptune ed esegue un attraversamento Gremlin.

Segui queste istruzioni da un'istanza Amazon EC2 nello stesso cloud privato virtuale (VPC) dell'istanza database Neptune.

Prima di iniziare, esegui queste attività:

- Verificare che Node.js versione 8.11 o successiva sia installato. In caso contrario, scaricare e installare Node.js dal sito Web Nodejs.org.

Per connettersi a Neptune tramite Node.js

1. Installare il pacchetto `gremlin-javascript` immettendo quanto segue:

```
npm install gremlin
```

2. Creare un file denominato `gremlinexample.js` e aprirlo in un editor di testo.
3. Copiare quanto segue nel file `gremlinexample.js`. Sostituisci *`your-neptune-endpoint`*con l'indirizzo della tua istanza DB Neptune.

   Per informazioni su come trovare l'indirizzo dell'istanza database Neptune, consulta la sezione Connessione agli endpoint Amazon Neptune.

```
const gremlin = require('gremlin');
const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;
const Graph = gremlin.structure.Graph;

dc = new DriverRemoteConnection('wss://your-neptune-endpoint:8182/gremlin',{});

const graph = new Graph();
const g = graph.traversal().withRemote(dc);

g.V().limit(1).count().next().
    then(data => {
        console.log(data);
        dc.close();
    }).catch(error => {
        console.log('ERROR', error);
        dc.close();
    });
```

4. Immettere il comando seguente per eseguire l'esempio:

```
node gremlinexample.js
```

L'esempio precedente restituisce il conteggio di un singolo vertice nel grafo utilizzando l'attraversamento `g.V().limit(1).count().next()`. Per eseguire query per qualcos'altro, sostituirla con un altro attraversamento Gremlin con uno dei metodi finali appropriati.

> **Note**
>
> La parte finale della query Gremlin, `next()` è obbligatoria per inviare l'attraversamento al server per la valutazione. Se non includi quel metodo o un altro metodo equivalente, la query non viene inviata all'istanza database Neptune.

I metodi riportati sotto inviano la query all'istanza database Neptune:

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

Utilizza `next()` se hai bisogno che i risultati della query vengano serializzati e restituiti oppure `iterate()` in caso contrario.

> **Important**
>
> Questo è un esempio Node.js standalone. Se hai intenzione di eseguire codice come questo in una AWS Lambda funzione, consulta Esempi di funzione Lambda per i dettagli sull'utilizzo JavaScript efficiente in una funzione Neptune Lambda.

Utilizzo di Go per connettersi a un'istanza database Neptune

Se puoi, usa sempre l'ultima versione del client Apache TinkerPop Go Gremlin, gremlingo, supportata dalla versione del tuo motore. Le nuove versioni contengono numerose correzioni di bug che migliorano la stabilità, le prestazioni e la fruibilità del client.

La `gremlingo` versione da usare in genere si allinea alle TinkerPop versioni descritte nella tabella per il client Java Gremlin.

> **Note**
>
> Le versioni di gremlingo 3.5.x sono retrocompatibili con le versioni 3.4.x purché si utilizzino solo le TinkerPop funzionalità 3.4.x nelle query Gremlin che si scrivono.

La sezione seguente illustra come eseguire un esempio Go che si connette a un'istanza database Amazon Neptune ed esegue un attraversamento Gremlin.

Segui queste istruzioni da un'istanza Amazon EC2 nello stesso cloud privato virtuale (VPC) dell'istanza database Neptune.

Prima di iniziare, esegui queste attività:

- Scarica e installa Go 1.17 o versioni successive dal sito Web go.dev.

Per connettersi a Neptune tramite Go

1. Partendo da una directory vuota, inizializza un nuovo modulo Go:

   ```
   go mod init example.com/gremlinExample
   ```

2. Aggiungi gremlin-go come dipendenza del nuovo modulo:

   ```
   go get github.com/apache/tinkerpop/gremlin-go/v3/driver
   ```

3. Crea un file denominato `gremlinExample.go`, quindi aprilo in un editor di testo.
4. Copia quanto segue nel file `gremlinExample.go`, sostituendo *`(your neptune endpoint)`* con l'indirizzo della tua istanza database Neptune:

   ```
   package main
   ```

```
import (
  "fmt"
  gremlingo "github.com/apache/tinkerpop/gremlin-go/v3/driver"
)

func main() {
  // Creating the connection to the server.
  driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection("wss://(your
 neptune endpoint):8182/gremlin",
    func(settings *gremlingo.DriverRemoteConnectionSettings) {
      settings.TraversalSource = "g"
    })
  if err != nil {
    fmt.Println(err)
    return
  }
  // Cleanup
  defer driverRemoteConnection.Close()

  // Creating graph traversal
  g := gremlingo.Traversal_().WithRemote(driverRemoteConnection)

  // Perform traversal
  results, err := g.V().Limit(2).ToList()
  if err != nil {
    fmt.Println(err)
    return
  }
  // Print results
  for _, r := range results {
    fmt.Println(r.GetString())
  }
}
```

Note

Il formato del certificato TLS Neptune non è attualmente supportato su Go 1.18+ con macOS e potrebbe restituire un errore 509 quando si tenta di avviare una connessione. Per i test locali, questo può essere ignorato aggiungendo "crypto/tls" alle importazioni e modificando le impostazioni `DriverRemoteConnection` come segue:

```
// Creating the connection to the server.
driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection("wss://
your-neptune-endpoint:8182/gremlin",
  func(settings *gremlingo.DriverRemoteConnectionSettings) {
      settings.TraversalSource = "g"
      settings.TlsConfig = &tls.Config{InsecureSkipVerify: true}
  })
```

5. Immettere il comando seguente per eseguire l'esempio:

```
go run gremlinExample.go
```

La query Gremlin alla fine di questo esempio restituisce i vertici (`g.V().Limit(2)`) in una sezione. Questa sezione viene quindi ripetuta e stampata con la funzione standard `fmt.Println`.

> **Note**
>
> La parte finale della query Gremlin, `ToList()` è obbligatoria per inviare l'attraversamento al server per la valutazione. Se non includi quel metodo o un altro metodo equivalente, la query non viene inviata all'istanza database Neptune.

I metodi riportati sotto inviano la query all'istanza database Neptune:

- `ToList()`
- `ToSet()`
- `Next()`
- `GetResultSet()`
- `Iterate()`

L'esempio precedente restituisce i primi due vertici del grafo utilizzando l'attraversamento `g.V().Limit(2).ToList()`. Per eseguire query per qualcos'altro, sostituirla con un altro attraversamento Gremlin con uno dei metodi finali appropriati.

Hint di query Gremlin

Puoi usare gli hint di query per specificare le strategie di ottimizzazione e valutazione per una particolare query Gremlin in Amazon Neptune.

Gli hint di query vengono specificati aggiungendo un passaggio `withSideEffect` alla query con la seguente sintassi.

```
g.withSideEffect(hint, value)
```

- hint: identifica il tipo di hint da applicare.
- value: determina il comportamento dell'aspetto del sistema in esame.

Ad esempio, di seguito è indicato come includere un hint `repeatMode` in un attraversamento Gremlin.

> **Note**
>
> Tutti gli hint di query Gremlin sono preceduti da `Neptune#`.

```
g.withSideEffect('Neptune#repeatMode',
 'DFS').V("3").repeat(out()).times(10).limit(1).path()
```

La query precedente indica al motore Neptune di attraversare il grafo Depth First (`DFS`) anziché il grafo Neptune predefinito, Breadth First (`BFS`).

Nelle seguenti sezioni sono fornite ulteriori informazioni sugli hint di query disponibili e sul relativo utilizzo.

Argomenti

Hint di query Gremlin repeatMode

L'hint di query Neptune `repeatMode` specifica come il motore Neptune valuta il passaggio `repeat()` in un attraversamento Gremlin: breadth first, depth first o chunked depth first.

La modalità di valutazione del passaggio `repeat()` è importante quando si usa per trovare o seguire un percorso, anziché ripetere un passaggio per un numero limitato di volte.

Sintassi

L'hint di query `repeatMode` si specifica aggiungendo un passaggio `withSideEffect` alla query.

```
g.withSideEffect('Neptune#repeatMode', 'mode').gremlin-traversal
```

> **Note**
>
> Tutti gli hint di query Gremlin sono preceduti da `Neptune#`.

Modalità disponibili

- `BFS`

  Breadth-First Search

  Modalità di esecuzione predefinita del passaggio `repeat()`. In tal modo si ottengono tutti i nodi di pari livello prima di procedere lungo il percorso.

  Questa versione richiede molta memoria e le frontiere possono diventare molto grandi. Esiste un rischio elevato che la query esaurisca la memoria e venga annullata dal motore Neptune. Questo caso si applica più strettamente ad altre implementazioni Gremlin.
- `DFS`

  Depth-First Search

  Segue ogni percorso alla profondità massima prima di passare alla soluzione successiva.

  Questa soluzione utilizza meno memoria. Può fornire prestazioni migliori in situazioni quali la ricerca di un singolo percorso partendo da un hop multiplo.
- `CHUNKED_DFS`

Chunked Depth-First Search

Un approccio ibrido che esplora il grafo depth-first in blocchi di 1.000 nodi, invece di 1 nodo (DFS) o tutti i nodi (BFS).

Il motore Neptune ottiene fino a 1.000 nodi a ogni livello prima di avanzare nel percorso.

Questo è un approccio equilibrato tra velocità e utilizzo della memoria.

È utile anche se vuoi usare BFS, ma la query sta usando troppa memoria.

Esempio

La sezione seguente descrive l'effetto della modalità di ripetizione su un attraversamento Gremlin.

In Neptune la modalità predefinita per il passaggio `repeat()` è eseguire una strategia di esecuzione breadth-first (BFS) per tutti gli attraversamenti.

Nella maggior parte dei casi, l' TinkerGraph implementazione utilizza la stessa strategia di esecuzione, ma in alcuni casi altera l'esecuzione di un attraversamento.

Ad esempio, l' TinkerGraph implementazione modifica la seguente query.

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

Il passaggio `repeat()` in questo attraversamento sarà "svolto" nell'attraversamento seguente, che risulta in una strategia depth-first (DFS).

```
g.V(<id>).out().out().out().out().out().out().out().out().out().out().limit(1).path()
```

> **Important**
>
> Il motore di query Neptune non esegue l'operazione automaticamente.

Breadth-first (BFS) è la strategia di esecuzione predefinita ed è simile nella maggior parte dei casi. TinkerGraph Tuttavia, vi sono alcuni casi in cui sono preferibili delle strategie depth-first (DFS).

BFS (impostazione predefinita)

Breadth-first (BFS) è la strategia di esecuzione predefinita per l'operatore `repeat()`.

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

Il motore Neptune esplora le prime frontiere a nove hop prima di trovare una soluzione a dieci hop. Questa soluzione è efficace in molti casi, come per una query del percorso più breve.

Tuttavia, per l'esempio precedente l'attraversamento sarebbe molto più veloce usando la modalità depth-first (DFS) per l'operatore `repeat()`.

DFS

La seguente query utilizza la modalità depth-first (DFS) per l'operatore `repeat()`.

```
g.withSideEffect("Neptune#repeatMode", "DFS").V("3").repeat(out()).times(10).limit(1)
```

Ogni singola soluzione viene seguita fino alla profondità massima prima di esplorare la soluzione successiva.

Hint di query Gremlin noReordering

Quando invii un attraversamento Gremlin, il motore di query Neptune esamina la struttura dell'attraversamento e riordina le parti della query, tentando di ridurre al minimo la quantità di lavoro richiesta per la valutazione e il tempo di risposta della query. Ad esempio, un attraversamento con più vincoli, come più passaggi `has()`, in genere non viene valutato nell'ordine specificato. Viene invece riordinato dopo che la query viene controllata con analisi statica.

Il motore di query Neptune tenta di identificare quale vincolo è più selettivo e lo esegue per primo. Questa soluzione spesso produce prestazioni migliori, ma l'ordine in cui Neptune sceglie di valutare la query potrebbe non essere sempre ottimale.

Se conosci le caratteristiche esatte dei dati e desideri indicare manualmente l'ordine di esecuzione della query, puoi utilizzare l'hint di query Neptune `noReordering` per specificare che l'attraversamento deve essere valutato nell'ordine specificato.

Sintassi

L'hint di query `noReordering` si specifica aggiungendo un passaggio `withSideEffect` alla query.

```
g.withSideEffect('Neptune#noReordering', true or false).gremlin-traversal
```

> **Note**
>
> Tutti gli hint di query Gremlin sono preceduti da `Neptune#`.

Valori disponibili

- `true`
- `false`

Hint di query Gremlin typePromotion

Quando invii un attraversamento Gremlin che filtra in base a un valore o intervallo numerico, il motore di query Neptune deve normalmente utilizzare la promozione dei tipi quando esegue la query. Ciò significa che deve esaminare i valori di ogni tipo che potrebbe contenere il valore in base al quale stai filtrando.

Ad esempio, se filtri per valori pari a 55, il motore deve cercare numeri interi pari a 55, numeri interi long pari a 55L, numeri a virgola mobile pari a 55.0 e così via. Ogni promozione di tipo richiede una ricerca aggiuntiva nell'archiviazione, che può far sì che una query apparentemente semplice richieda un tempo inaspettatamente lungo per essere completata.

Supponiamo che tu stia cercando tutti i vertici con una proprietà relativa all'età del cliente maggiore di 5:

```
g.V().has('customerAge', gt(5))
```

Per eseguire tale attraversamento in modo completo, Neptune deve espandere la query per esaminare ogni tipo numerico a cui potrebbe essere promosso il valore per cui stai eseguendo la query. In questo caso, il filtro `gt` deve essere applicato a qualsiasi numero intero superiore a 5, a qualsiasi numero long superiore a 5L, a qualsiasi numero a virgola mobile superiore a 5.0 e a qualsiasi valore double superiore a 5.0. Poiché ognuna di queste promozioni di tipo richiede una ricerca aggiuntiva nell'archiviazione, quando esegui [API Gremlin `profile`] per questa query vedrai più filtri per ogni filtro numerico e il completamento richiederà molto più tempo di quanto potresti aspettarti.

Spesso la promozione dei tipi non è necessaria perché si sa in anticipo che è necessario trovare solo valori di un tipo specifico. In questo caso, puoi velocizzare notevolmente le query utilizzando l'hint di query `typePromotion` per disattivare la promozione dei tipi.

Sintassi

L'hint di query `typePromotion` si specifica aggiungendo un passaggio `withSideEffect` alla query.

```
g.withSideEffect('Neptune#typePromotion', true or false).gremlin-traversal
```

> **Note**
>
> Tutti gli hint di query Gremlin sono preceduti da `Neptune#`.

Valori disponibili

- `true`
- `false`

Per disattivare la promozione dei tipi per la query precedente, puoi usare:

```
g.withSideEffect('Neptune#typePromotion', false).V().has('customerAge', gt(5))
```

Hint di query Gremlin useDFE

Utilizzare questo hint di query per abilitare l'uso del motore DFE per l'esecuzione della query. Per impostazione predefinita, Neptune non utilizza il motore DFE senza che questo hint di query sia impostato su `true`, poiché il parametro di istanza neptune_dfe_query_engine è impostato su `viaQueryHint`. Se imposti il parametro di istanza su `enabled`, il motore DFE viene utilizzato per tutte le query ad eccezione di quelle con l'hint di query `useDFE` impostato su `false`.

Esempio di abilitazione del motore DFE per una query:

```
g.withSideEffect('Neptune#useDFE', true).V().out()
```

Hint di query Gremlin per l'utilizzo della cache dei risultati

I seguenti hint di query possono essere utilizzati quando è abilitata la cache dei risultati delle query.

Hint di query Gremlin **enableResultCache**

L'hint di query `enableResultCache` con valore `true` fa sì che i risultati della query vengano restituiti dalla cache se sono già stati memorizzati nella cache. In caso contrario, restituisce i nuovi risultati e li memorizza nella cache fino a quando non vengono cancellati dalla cache. Per esempio:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Successivamente, è possibile accedere ai risultati memorizzati nella cache eseguendo di nuovo esattamente la stessa query.

Se il valore di questo hint di query è `false` o se non è presente, i risultati della query non vengono memorizzati nella cache. Tuttavia, impostando il valore su `false` non si cancellano i risultati esistenti memorizzati nella cache. Per cancellare i risultati memorizzati nella cache, usare l'hint `invalidateResultCache` o `invalidateResultCachekey`.

Hint di query Gremlin **enableResultCacheWithTTL**

L'hint di query `enableResultCacheWithTTL` restituisce anche i risultati memorizzati nella cache, se presenti, senza influire sul TTL dei risultati già presenti nella cache. Se attualmente non sono presenti risultati nella cache, la query restituisce nuovi risultati e li memorizza nella cache per il time to live (TTL) specificato dall'hint di query `enableResultCacheWithTTL`. Il time to live è specificato in secondi. Ad esempio, la seguente query specifica un time to live di sessanta secondi:

```
g.with('Neptune#enableResultCacheWithTTL', 60)
 .V().has('genre','drama').in('likes')
```

Prima che i 60 secondi time-to-live siano trascorsi, puoi usare la stessa query (quig`.V().has('genre','drama').in('likes')`) con il suggerimento `enableResultCache` o la `enableResultCacheWithTTL` query per accedere ai risultati memorizzati nella cache.

> **Note**
>
> Il time to live specificato con `enableResultCacheWithTTL` non influisce sui risultati che sono già stati memorizzati nella cache.

- Se i risultati sono stati precedentemente memorizzati nella cache utilizzando `enableResultCache`, la cache deve essere cancellata in modo esplicito prima che `enableResultCacheWithTTL` generi nuovi risultati e li memorizzi nella cache per il TTL specificato.
- Se i risultati sono stati precedentemente memorizzati nella cache utilizzando `enableResultCachewithTTL`, il TTL precedente deve scadere prima che `enableResultCacheWithTTL` generi nuovi risultati e li memorizzi nella cache per il TTL specificato.

Una volta trascorso il time to live, i risultati della query memorizzati nella cache vengono cancellati e un'istanza successiva della stessa query restituisce nuovi risultati. Se `enableResultCacheWithTTL` è associato alla query successiva, i nuovi risultati vengono memorizzati nella cache con il TTL specificato.

Hint di query Gremlin **`invalidateResultCacheKey`**

L'hint di query `invalidateResultCacheKey` può accettare il valore `true` o `false`. Il valore `true` fa sì che i risultati memorizzati nella cache per la query a cui `invalidateResultCacheKey` è associato vengano cancellati. Ad esempio, l'esempio seguente fa sì che i risultati memorizzati nella cache per la chiave di query `g.V().has('genre','drama').in('likes')` vengano cancellati:

```
g.with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

La query di esempio precedente non fa sì che i nuovi risultati vengano memorizzati nella cache. È possibile includere `enableResultCache` (o `enableResultCacheWithTTL`) nella stessa query se si desidera memorizzare nella cache i nuovi risultati dopo aver cancellato quelli esistenti:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Hint di query Gremlin **`invalidateResultCache`**

L'hint di query `invalidateResultCache` può accettare il valore `true` o `false`. Il valore `true` fa sì che tutti i risultati nella cache dei risultati vengano cancellati. Per esempio:

```
g.with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

La query di esempio precedente non fa sì che i relativi risultati vengano memorizzati nella cache. È possibile includere `enableResultCache` (o `enableResultCacheWithTTL`) nella stessa query se si desidera memorizzare nella cache i nuovi risultati dopo aver cancellato completamente la cache esistente:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Hint di query Gremlin **`numResultsCached`**

L'hint di query `numResultsCached` può essere utilizzato solo con le query che contengono `iterate()` e specifica il numero massimo di risultati da memorizzare nella cache per la query a cui è associato. Si not che i risultati memorizzati nella cache quando è presente `numResultsCached` non vengono restituiti, ma solo memorizzati nella cache.

Ad esempio, la seguente query specifica che devono essere memorizzati nella cache fino a 100 risultati, ma nessuno di questi risultati memorizzati nella cache viene restituito:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').iterate()
```

È quindi possibile utilizzare una query come la seguente per recuperare un intervallo di risultati memorizzati nella cache (in questo caso i primi dieci):

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').range(0, 10)
```

Hint di query Gremlin **`noCacheExceptions`**

L'hint di query `noCacheExceptions` può accettare il valore `true` o `false`. Il valore `true` causa l'eliminazione di tutte le eccezioni relative alla cache dei risultati. Per esempio:

```
g.with('Neptune#enableResultCache', true)
```

```
  .with('Neptune#noCacheExceptions', true)
  .V().has('genre','drama').in('likes')
```

In particolare, elimina l'eccezione `QueryLimitExceededException`, che viene generato se i risultati di una query sono troppo grandi per essere inseriti nella cache dei risultati.

API di stato delle query Gremlin

Per ottenere lo stato delle query Gremlin, utilizza l'operazione HTTP `GET` o `POST` per effettuare una richiesta all'endpoint `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin/status`.

Parametri della richiesta di stato delle query Gremlin

- queryId (opzionale): ID di una query Gremlin in esecuzione. Viene mostrato solo lo stato della query specificata.
- includeWaiting (opzionale): restituisce lo stato di tutte le query in attesa.

  Normalmente, nella risposta sono incluse solo le query in esecuzione, ma quando viene specificato il parametro `includeWaiting`, viene restituito anche lo stato di tutte le query in attesa.

Sintassi della risposta di stato delle query Gremlin

```
{
  "acceptedQueryCount": integer,
  "runningQueryCount": integer,
  "queries": [
    {
      "queryId":"guid",
      "queryEvalStats":
        {
          "waited": integer,
          "elapsed": integer,
          "cancelled": boolean
        },
      "queryString": "string"
    }
  ]
}
```

Valori della risposta di stato delle query Gremlin

- acceptedQueryCount— Il numero di interrogazioni che sono state accettate ma non ancora completate, incluse le interrogazioni in coda.
- runningQueryCount— Il numero di interrogazioni Gremlin attualmente in esecuzione.
- queries: elenco delle query Gremlin correnti.
- queryId: ID GUID della query. Neptune assegna automaticamente questo valore ID a ogni query oppure è possibile assegnare un ID personalizzato (consulta Inserimento di un ID personalizzato in una query Neptune Gremlin o SPARQL).
- queryEvalStats— Statistiche per questa interrogazione.
- subqueries: numero di sottoquery in questa query.
- elapsed: numero di millisecondi in cui la query è stata eseguita finora.
- cancelled: il valore True indica che la query è stata annullata.
- queryString: la query inviata. Questa è troncata a 1024 caratteri nel caso in cui sia più lunga.
- waited: indica il tempo di attesa della query, in millisecondi.

Esempio di stato di una query Gremlin

Di seguito è riportato un esempio del comando di stato che utilizza `curl` e l'operazione HTTP `GET`.

```
curl https://your-neptune-endpoint:port/gremlin/status
```

Questo output mostra una sola query in esecuzione.

```
{
  "acceptedQueryCount":9,
  "runningQueryCount":1,
  "queries": [
    {
      "queryId":"fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
      "queryEvalStats":
        {
          "waited": 0,
          "elapsed": 23,
          "cancelled": false
        },
      "queryString": "g.V().out().count()"
```

```
    }
  ]
}
```

Annullamento delle query Gremlin

Per ottenere lo stato delle query Gremlin, utilizza l'operazione HTTP `GET` o `POST` per effettuare una richiesta all'endpoint `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin/status`.

Parametri della richiesta di annullamento delle query Gremlin

- cancelQuery: obbligatorio per l'annullamento. Il parametro non dispone di un valore corrispondente.
- queryId: ID della query Gremlin in esecuzione da annullare.

Esempio di annullamento di una query Gremlin

Di seguito è riportato un esempio del comando `curl` per annullare una query.

```
curl https://your-neptune-endpoint:port/gremlin/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=fb34cd3e-f37c-4d12-9cf2-03bb741bf54f"
```

L'avvenuto annullamento restituisce il codice HTTP `200` impostato su OK.

Supporto delle sessioni basate su script Gremlin

È possibile utilizzare le sessioni Gremlin con transazioni implicite in Amazon Neptune. Per informazioni sulle sessioni Gremlin, vedere Considering Sessions nella documentazione di TinkerPop Apache. Le sezioni seguenti descrivono come utilizzare le sessioni Gremlin con Java.

> **Note**
>
> Questa funzionalità è disponibile a partire dal rilascio 1.0.1.0.200463.0 del motore Neptune. A partire dalla versione 1.1.1.0 TinkerPop e 3.5.2 del motore Neptune, puoi anche usare. Transazioni Gremlin

> **Important**
>
> Attualmente il periodo di tempo massimo in cui Neptune può mantenere aperta una sessione basata su script è pari a 10 minuti. Se non si chiude una sessione prima di questo tempo, la sessione scade e tutto il contenuto viene sottoposto a rollback.

Argomenti

Sessioni Gremlin sulla console Gremlin

Se crei una connessione remota sulla console Gremlin senza il parametro `session`, la connessione remota viene creata in modalità senza sessioni . In questa modalità, ogni richiesta inviata al server viene considerata come una transazione completa in sé e nessuno stato viene salvato tra le richieste. Se una richiesta ha esito negativo, viene eseguito il rollback solo di quella richiesta.

Se si crea una connessione remota che utilizza il parametro `session`, si crea una sessione basata su script che dura finché non si chiude la connessione remota. Ogni sessione è identificata da un UUID univoco che la console genera e restituisce.

Di seguito è riportato un esempio di una chiamata della console che crea una sessione. Dopo aver inviato le query, un'altra chiamata chiude la sessione ed esegue il commit delle query.

> **Note**
>
> Il client Gremlin deve essere sempre chiuso per rilasciare le risorse lato server.

```
gremlin> :remote connect tinkerpop.server conf/neptune-remote.yaml session
  . . .
  . . .
gremlin> :remote close
```

Per ulteriori informazioni ed esempi, consulta Sessions nella documentazione. TinkerPop

Tutte le query eseguite durante una sessione formano una singola transazione di cui non viene eseguito il commit finché tutte le query non vanno a buon fine e la connessione remota non viene chiusa. Se una query non riesce o se non si chiude la connessione entro la durata massima della sessione supportata da Neptune, non viene eseguito il commit della transazione di sessione e viene eseguito il rollback di tutte le query in essa contenute

Sessioni Gremlin nella variante del linguaggio Gremlin

Nel Gremlin Language Variant, è necessario creare un oggetto `SessionedClient` per inviare più query in una singola transazione, come descritto nell'esempio seguente.

```
try {                              // line 1
  Cluster cluster = Cluster.open();                    // line 2
  Client client = cluster.connect("sessionName");      // line 3
   ...
   ...
} finally {
  // Always close. If there are no errors, the transaction is committed; otherwise,
 it's rolled back.
  client.close();
}
```

La riga 3 nell'esempio precedente crea l'oggetto `SessionedClient` in base alle opzioni di configurazione impostate per il cluster in questione. La stringa *`sessionName`* che si invia al metodo di connessione diventa il nome univoco della sessione. Per evitare collisioni, utilizzare un UUID per il nome.

Il client avvia una transazione di sessione quando viene inizializzato. Il commit di tutte le query che esegui durante il modulo di sessione viene eseguito solo quando chiami `client.close( )`. Ancora una volta, se una singola query non riesce o se non si chiude la connessione entro la durata massima supportata da Neptune, la transazione della sessione non riesce e viene eseguito il rollback di tutte le query in essa contenute.

> **Note**
>
> Il client Gremlin deve essere sempre chiuso per rilasciare le risorse lato server.

```
GraphTraversalSource g = traversal().withRemote(conn);
```

```
Transaction tx = g.tx();

// Spawn a GraphTraversalSource from the Transaction.
// Traversals spawned from gtx are executed within a single transaction.
GraphTraversalSource gtx = tx.begin();
try {
  gtx.addV('person').iterate();
  gtx.addV('software').iterate();

  tx.commit();
} finally {
    if (tx.isOpen()) {
        tx.rollback();
    }
}
```

Transazioni Gremlin in Neptune

Esistono diversi contesti in cui vengono eseguite le transazioni Gremlin. Quando si lavora con Gremlin è importante comprendere il contesto in cui si lavora e quali sono le relative implicazioni:

- **`Script-based`**: le richieste vengono effettuate utilizzando stringhe Gremlin basate su testo, come queste:
  - Utilizzando il driver Java e `Client.submit(`*`string`*`)`.
  - Utilizzando la console Gremlin e `:remote connect`.
  - Utilizzando l'API HTTP.
- **`Bytecode-based`**: le richieste vengono effettuate utilizzando il bytecode Gremlin serializzato tipico delle varianti del linguaggio Gremlin (GLV).

  Ad esempio, utilizzando il driver Java, `g = traversal().withRemote(...)`.

Per entrambi i contesti sopra descritti, esiste il contesto aggiuntivo della richiesta inviata come senza sessione o come associata a una sessione.

> **Note**
>
> Le transazioni Gremlin devono sempre essere sottoposte a commit o rollback, in modo che le risorse lato server possano essere rilasciate.

Richieste senza sessione

Quando è senza sessione, una richiesta equivale a una singola transazione.

Per gli script, l'implicazione è che una o più istruzioni Gremlin inviate in una singola richiesta verranno sottoposte a commit o rollback come singola transazione. Per esempio:

```
Cluster cluster = Cluster.open();
Client client = cluster.connect(); // sessionless
// 3 vertex additions in one request/transaction:
client.submit("g.addV();g.addV();g.addV()").all().get();
```

Per il bytecode, viene effettuata una richiesta senza sessione per ogni attraversamento generato ed eseguito da g:

```
GraphTraversalSource g = traversal().withRemote(...);

// 3 vertex additions in three individual requests/transactions:
g.addV().iterate();
g.addV().iterate();
g.addV().iterate();

// 3 vertex additions in one single request/transaction:
g.addV().addV().addV().iterate();
```

Richieste associate a una sessione

Se associate a una sessione, è possibile applicare più richieste nel contesto di una singola transazione.

Per gli script, l'implicazione è che non è necessario concatenare tutte le operazioni del grafo in un unico valore di stringa incorporato:

```
Cluster cluster = Cluster.open();
Client client = cluster.connect(sessionName); // session
try {
    // 3 vertex additions in one request/transaction:
    client.submit("g.addV();g.addV();g.addV()").all().get();
} finally {
    client.close();
}
```

```
try {
    // 3 vertex additions in three requests, but one transaction:
    client.submit("g.addV()").all().get(); // starts a new transaction with the same
 sessionName
    client.submit("g.addV()").all().get();
    client.submit("g.addV()").all().get();
} finally {
    client.close();
}
```

Per quanto riguarda il bytecode, in seguito TinkerPop `3.5.x`, la transazione può essere controllata in modo esplicito e la sessione gestita in modo trasparente. Le varianti del linguaggio Gremlin (GLV) supportano la sintassi `tx()` di Gremlin per eseguire le operazioni di `commit()` o `rollback()` su una transazione come segue:

```
GraphTraversalSource g = traversal().withRemote(conn);

Transaction tx = g.tx();

// Spawn a GraphTraversalSource from the Transaction.
// Traversals spawned from gtx are executed within a single transaction.
GraphTraversalSource gtx = tx.begin();
try {
    gtx.addV('person').iterate();
    gtx.addV('software').iterate();

    tx.commit();
} finally {
    if (tx.isOpen()) {
        tx.rollback();
    }
}
```

Sebbene l'esempio precedente sia scritto in Java, è possibile utilizzare questa sintassi `tx()` anche in Python, Javascript e .NET.

> **Warning**
>
> Le query di sola lettura senza sessione vengono eseguite con isolamento SNAPSHOT, mentre le query di sola lettura eseguite all'interno di una transazione esplicita vengono eseguite con isolamento SERIALIZABLE. Le query di sola lettura eseguite con isolamento

`SERIALIZABLE` comportano un sovraccarico maggiore e possono bloccarsi o essere bloccate da scritture simultanee, a differenza di quelle eseguite con isolamento `SNAPSHOT`.

Utilizzo dell'API Gremlin con Amazon Neptune

> **Note**
>
> Amazon Neptune non supporta la proprietà `bindings`.

Tutte le richieste HTTPS Gremlin utilizzano un singolo endpoint: `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin`. Tutte le connessioni Neptune devono utilizzare HTTPS.

È possibile collegare la console Gremlin a un grafico di Neptune direttamente tramite. WebSockets

Per ulteriori informazioni sulla connessione all'endpoint di Gremlin, consulta Accesso al grafo Neptune con Gremlin.

L'implementazione di Gremlin in Amazon Neptune prevede dettagli specifici e differenze da prendere in considerazione. Per ulteriori informazioni, consulta Conformità agli standard Gremlin in Amazon Neptune.

Per informazioni sul linguaggio Gremlin e sui suoi attraversamenti, consulta The Traversal nella documentazione di Apache. TinkerPop

Memorizzazione nella cache dei risultati delle query con Gremlin in Amazon Neptune

A partire dal rilascio 1.0.5.1 del motore, Amazon Neptune supporta una cache dei risultati per le query Gremlin.

È possibile abilitare la cache dei risultati delle query e quindi utilizzare un hint di query per memorizzare nella cache i risultati di una query di sola lettura Gremlin.

Qualsiasi riesecuzione della query recupera quindi i risultati memorizzati nella cache con bassa latenza e senza costi di I/O, purché siano ancora nella cache. Funziona per le query inviate sia su un endpoint HTTP che tramite Websocket, sia come codice byte che sotto forma di stringa.

> **Note**
>
> Le query inviate all'endpoint del profilo non vengono memorizzate nella cache anche quando la cache delle query è abilitata.

È possibile controllare il comportamento della cache dei risultati delle query di Neptune in diversi modi. Per esempio:

- È possibile ottenere risultati memorizzati nella cache paginati, in blocchi.
- È possibile specificare il (TTL) per le query specificate. time-to-live
- È possibile cancellare la cache per query specifiche.
- È possibile cancellare l'intera cache.
- È possibile impostare la possibilità di ricevere una notifica se i risultati superano la dimensione della cache.

La cache viene gestita utilizzando una politica least-recently-used (LRU), il che significa che una volta che lo spazio assegnato alla cache è pieno, i least-recently-used risultati vengono rimossi per liberare spazio quando i nuovi risultati vengono memorizzati nella cache.

> **Important**
>
> La cache dei risultati delle query non è disponibile per i tipi di istanza `t3.medium` o `t4.medium`.

Abilitazione della cache dei risultati delle query in Neptune

Per abilitare la cache dei risultati delle query in Neptune, usare la console per impostare il parametro dell'istanza database `neptune_result_cache` su `1` (abilitato).

Una volta abilitata la cache dei risultati, Neptune riserva una parte della memoria corrente per memorizzare nella cache i risultati delle query. Più grande è il tipo di istanza utilizzato e maggiore è la memoria disponibile, maggiore sarà la quantità di memoria che Neptune riserverà per la cache.

Se la memoria cache dei risultati si riempie, Neptune least-recently-used elimina automaticamente i risultati (LRU) memorizzati nella cache per far posto a quelli nuovi.

È possibile verificare lo stato corrente della cache dei risultati utilizzando il comando Stato dell'istanza.

Utilizzo degli hint per memorizzare nella cache i risultati delle query

Una volta abilitata la cache dei risultati delle query, è possibile utilizzare gli hint di query per controllare la memorizzazione nella cache delle query. Tutti gli esempi seguenti si applicano allo stesso attraversamento della query, vale a dire:

```
g.V().has('genre','drama').in('likes')
```

Uso di **`enableResultCache`**

Con la cache dei risultati delle query abilitata, è possibile memorizzare nella cache i risultati di una query Gremlin utilizzando l'hint di query `enableResultCache`, come segue:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Neptune restituisce quindi i risultati della query e li memorizza nella cache. Successivamente, è possibile accedere ai risultati memorizzati nella cache eseguendo di nuovo esattamente la stessa query:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

La chiave della cache che identifica i risultati memorizzati nella cache è la stringa di query stessa, vale a dire:

```
g.V().has('genre','drama').in('likes')
```

Uso di **`enableResultCacheWithTTL`**

È possibile specificare per quanto tempo i risultati delle query devono essere memorizzati nella cache utilizzando l'hint di query `enableResultCacheWithTTL`. Ad esempio, la seguente query specifica che i risultati della query devono scadere dopo 120 secondi:

```
g.with('Neptune#enableResultCacheWithTTL', 120)
```

```
  .V().has('genre','drama').in('likes')
```

Anche in questo caso, la chiave della cache che identifica i risultati memorizzati nella cache è la stringa di query di base:

```
g.V().has('genre','drama').in('likes')
```

E anche in questo caso, è possibile accedere ai risultati memorizzati nella cache usando tale stringa di query con l'hint di query `enableResultCache`:

```
g.with('Neptune#enableResultCache', true)
  .V().has('genre','drama').in('likes')
```

Se sono trascorsi 120 o più secondi da quando i risultati sono stati memorizzati nella cache, quella query restituirà nuovi risultati e li memorizzerà nella cache, senza alcun risultato. time-to-live

È inoltre possibile accedere ai risultati memorizzati nella cache inviando nuovamente la stessa query con l'hint di query `enableResultCacheWithTTL`. Per esempio:

```
g.with('Neptune#enableResultCacheWithTTL', 140)
  .V().has('genre','drama').in('likes')
```

Finché non sono trascorsi 120 secondi (ovvero il TTL attualmente in vigore), questa nuova query che utilizza l'hint di query `enableResultCacheWithTTL` restituisce i risultati memorizzati nella cache. Dopo 120 secondi, restituirà nuovi risultati e li memorizzerà nella cache per 140 secondi. time-to-live

> **Note**
>
> Se i risultati di una chiave di query sono già memorizzati nella cache, la stessa chiave di query con `enableResultCacheWithTTL` non genera nuovi risultati e non ha alcun effetto sui time-to-live risultati attualmente memorizzati nella cache.
>
> - Se i risultati sono stati precedentemente memorizzati nella cache utilizzando `enableResultCache`, la cache deve essere cancellata prima che `enableResultCacheWithTTL` generi nuovi risultati e li memorizzi nella cache per il TTL specificato.
> - Se i risultati sono stati precedentemente memorizzati nella cache utilizzando `enableResultCachewithTTL`, il TTL precedente deve scadere prima che

`enableResultCacheWithTTL` generi nuovi risultati e li memorizzi nella cache per il TTL specificato.

Uso di **`invalidateResultCacheKey`**

È possibile utilizzare l'hint di query `invalidateResultCacheKey` per cancellare i risultati memorizzati nella cache per una determinata query. Per esempio:

```
g.with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Questa query cancella la cache per la chiave di query, `g.V().has('genre','drama').in('likes')`, e restituisce nuovi risultati per la query.

È inoltre possibile combinare `invalidateResultCacheKey` con `enableResultCache` o `enableResultCacheWithTTL`. Ad esempio, la seguente query cancella i risultati correnti memorizzati nella cache, memorizza nella cache i nuovi risultati e li restituisce:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Uso di **`invalidateResultCache`**

È possibile utilizzare l'hint di query `invalidateResultCache` per cancellare tutti i risultati memorizzati nella cache dei risultati della query. Per esempio:

```
g.with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Questa query cancella l'intera cache dei risultati e restituisce nuovi risultati per la query.

È inoltre possibile combinare `invalidateResultCache` con `enableResultCache` o `enableResultCacheWithTTL`. Ad esempio, la seguente query cancella l'intera cache dei risultati, memorizza nella cache i nuovi risultati per questa query e li restituisce:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCache', true)
```

```
  .V().has('genre','drama').in('likes')
```

Paginazione dei risultati delle query memorizzati nella cache

Supponiamo di aver già memorizzato nella cache un gran numero di risultati come questo:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Supponiamo ora di inviare la seguente query di intervallo:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes').range(0,10)
```

Neptune cerca innanzitutto la chiave completa della cache, vale a dire `g.V().has('genre','drama').in('likes').range(0,10)`. Se tale chiave non esiste, Neptune cerca se esiste una chiave per la stringa di query senza l'intervallo (vale a dire `g.V().has('genre','drama').in('likes')`). Quando trova la chiave, Neptune recupera i primi dieci risultati dalla relativa cache, come specificato dall'intervallo.

> **Note**
>
> Se si usa l'hint di query `invalidateResultCacheKey` con una query che ha un intervallo alla fine, Neptune cancella la cache per una query senza l'intervallo se non trova una corrispondenza esatta per la query con l'intervallo.

Uso di **`numResultsCached`** con **`.iterate()`**

Utilizzando l'hint di query `numResultsCached`, è possibile popolare la cache dei risultati senza restituire tutti i risultati memorizzati nella cache, il che può essere utile quando si preferisce paginare un numero elevato di risultati.

L'hint di query `numResultsCached` funziona solo con le query che terminano con `iterate()`.

Ad esempio, se si desidera memorizzare nella cache i primi 50 risultati della query di esempio:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
```

```
 .V().has('genre','drama').in('likes').iterate()
```

In questo caso la chiave di query nella cache è: `g.with("Neptune#numResultsCached", 50).V().has('genre','drama').in('likes')`. Ora è possibile recuperare i primi dieci risultati memorizzati nella cache con questa query:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').range(0, 10)
```

Inoltre, è possibile recuperare i dieci risultati successivi dalla query nel modo seguente:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').range(10, 20)
```

Non dimenticare di includere l'hint `numResultsCached`. È una parte essenziale della chiave di query e deve quindi essere presente per poter accedere ai risultati memorizzati nella cache.

Alcune cose da tenere a mente quando si utilizza **`numResultsCached`**:

- Il numero fornito con **`numResultsCached`** viene applicato alla fine della query. Ciò significa, ad esempio, che la seguente query memorizza effettivamente nella cache i risultati nell'intervallo `(1000, 1500)`:

  ```
  g.with("Neptune#enableResultCache", true)
   .with("Neptune#numResultsCached", 500)
   .V().range(1000, 2000).iterate()
  ```

- Il numero fornito con **`numResultsCached`** specifica il numero massimo di risultati da memorizzare nella cache. Ciò significa, ad esempio, che la seguente query memorizza effettivamente nella cache i risultati nell'intervallo `(1000, 2000)`:

  ```
  g.with("Neptune#enableResultCache", true)
   .with("Neptune#numResultsCached", 100000)
   .V().range(1000, 2000).iterate()
  ```

- I risultati memorizzati nella cache delle query che terminano con **`.range().iterate()`** hanno un proprio intervallo. Ad esempio, supponiamo di memorizzare nella cache i risultati utilizzando una query come questa:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).iterate()
```

Per recuperare i primi 100 risultati dalla cache, scrivere una query come questa:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).range(0, 100)
```

Questi cento risultati saranno equivalenti ai risultati della query di base nell'intervallo `(1000, 1100)`.

Chiavi della cache delle query utilizzate per individuare i risultati memorizzati nella cache

Dopo che i risultati di una query sono stati memorizzati nella cache, le query successive con la stessa chiave di cache delle query recuperano i risultati dalla cache anziché generarne di nuovi. La chiave di cache di una query viene valutata come segue:

1. Tutti gli hint di query relativi alla cache vengono ignorati, ad eccezione di `numResultsCached`.
2. Il passaggio finale `iterate()` viene ignorato.
3. Il resto della query viene ordinato in base alla rappresentazione del codice byte.

La stringa risultante viene confrontata con un indice dei risultati della query già presenti nella cache per determinare se esiste un riscontro nella cache per la query.

Ad esempio, prendiamo questa query:

```
g.withSideEffect('Neptune#typePromotion', false).with("Neptune#enableResultCache",
 true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').iterate()
```

Verrà memorizzata come versione in codice byte di questa:

```
g.withSideEffect('Neptune#typePromotion', false)
```

```
  .with("Neptune#numResultsCached", 50)
  .V().has('genre','drama').in('likes')
```

Eccezioni relative alla cache dei risultati

Se i risultati di una query che si sta cercando di memorizzare nella cache sono troppo grandi per essere contenuti nella memoria cache anche dopo aver rimosso tutto ciò che era precedentemente memorizzato nella cache, Neptune genera un errore `QueryLimitExceededException`. Non viene restituito alcun risultato e l'eccezione genera il seguente messaggio di errore:

```
The result size is larger than the allocated cache,
      please refer to results cache best practices for options to rerun the query.
```

È possibile eliminare questo messaggio utilizzando l'hint di query `noCacheExceptions`, come segue:

```
g.with('Neptune#enableResultCache', true)
  .with('Neptune#noCacheExceptions', true)
  .V().has('genre','drama').in('likes')
```

Creazione di upsert efficienti con i passaggi **`mergeV()`** e **`mergeE()`** di Gremlin

Un upsert (o inserimento condizionale) riutilizza un vertice o uno arco se già esiste oppure lo crea se non esiste. Upsert efficienti possono fare una differenza significativa nelle prestazioni delle query Gremlin.

Gli upsert consentono di scrivere operazioni di inserimento idempotenti: indipendentemente dal numero di volte in cui si esegue un'operazione di questo tipo, il risultato complessivo è lo stesso. Ciò è utile in scenari di scrittura altamente simultanei in cui modifiche simultanee alla stessa parte del grafo possono forzare il rollback di una o più transazioni con un'eccezione `ConcurrentModificationException`, rendendo quindi necessari nuovi tentativi.

Ad esempio, la seguente query esegue l'upsert di un vertice utilizzando l'oggetto `Map` fornito per cercare prima un vertice con `T.id` di `"v-1"`. Se il vertice viene trovato, viene restituito. Se non viene trovato, viene creato un vertice con tale valore `id` e tale proprietà tramite la clausola `onCreate`.

```
g.mergeV([(id):'v-1']).
  option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org'])
```

Upsert in batch per migliorare la velocità di trasmissione effettiva

Per scenari di scrittura con velocità di trasmissione effettiva elevata, è possibile concatenare i passaggi `mergeV()` e `mergeE()` per eseguire l'upsert di vertici e archi in batch. Il batching riduce il sovraccarico transazionale dell'upsert di un gran numero di vertici e archi. È quindi possibile migliorare ulteriormente la velocità di trasmissione effettiva eseguendo l'upsert di richieste batch in parallelo utilizzando più client.

Come regola generale, è consigliabile eseguire l'upsert di circa 200 record per richiesta batch. Un record è una singola etichetta o proprietà di un vertice o un arco. Un vertice con una singola etichetta e 4 proprietà, ad esempio, crea 5 record. Un arco con un'etichetta e una singola proprietà crea 2 record. Se si vuole eseguire l'upsert di batch di vertici, ciascuno con una singola etichetta e 4 proprietà, è necessario iniziare con una dimensione di batch di 40, perché `200 / (1 + 4) = 40`.

È possibile sperimentare con la dimensione del batch. 200 record per batch sono un buon punto di partenza, ma la dimensione ideale del batch può essere maggiore o minore a seconda del carico di lavoro. Tenere presente, tuttavia, che Neptune può limitare il numero complessivo di passaggi Gremlin per richiesta. Questo limite non è documentato, ma per sicurezza, cercare di assicurarsi che le richieste non contengano più di 1.500 passaggi Gremlin. Neptune può rifiutare richieste batch di grandi dimensioni con più di 1.500 passaggi.

Per aumentare la velocità di trasmissione effettiva, è possibile eseguire l'upsert di batch in parallelo utilizzando più client (vedi Creazione di scritture multithread Gremlin efficienti). Il numero di client deve essere uguale al numero di thread di lavoro sull'istanza di scrittura Neptune, che in genere è 2 volte il numero di vCPU sul server. Ad esempio, un'istanza `r5.8xlarge` ha 32 vCPU e 64 thread di lavoro. Per scenari di scrittura con velocità di trasmissione effettiva elevata che utilizzano un'istanza `r5.8xlarge`, è necessario utilizzare 64 client che scrivono upsert in batch su Neptune in parallelo.

Ogni client deve inviare una richiesta batch e attendere il completamento della richiesta prima di inviarne un'altra. Sebbene più client vengano eseguiti in parallelo, ogni singolo client invia le richieste in modo seriale. Ciò garantisce che il server riceva un flusso costante di richieste che occupano tutti i thread di lavoro senza sovraccaricare la coda delle richieste lato server (vedi Dimensionamento delle istanze database in un cluster database Neptune).

Cercare di evitare passaggi che generano più traverser

Quando viene eseguito un passaggio Gremlin, accetta un traverser in entrata ed emette uno o più traverser in uscita. Il numero di traverser emessi da un passaggio determina il numero di volte in cui viene eseguito il passaggio successivo.

In genere, quando si eseguono operazioni in batch, si desidera che ogni operazione, ad esempio l'upsert del vertice A, venga eseguita una sola volta, in modo che la sequenza di operazioni sia la seguente: upsert del vertice A, quindi upsert del vertice B, quindi upsert del vertice C e così via. Finché un passaggio crea o modifica un solo elemento, emette un solo traverser e i passaggi che rappresentano l'operazione successiva vengono eseguiti una sola volta. Se, invece, un'operazione crea o modifica più di un elemento, emette più traverser, che a loro volta fanno sì che i passaggi successivi vengano eseguiti più volte, una volta per ogni traverser emesso. Ciò può comportare l'esecuzione di operazioni aggiuntive non necessarie nel database e, in alcuni casi, la creazione di vertici, archi o valori di proprietà aggiuntivi indesiderati.

Un esempio di come le cose possano andare male è con una query come `g.V().addV()`. Questa semplice query aggiunge un vertice per ogni vertice trovato nel grafo, perché `V()` emette un traverser per ogni vertice del grafo e ognuno di questi traverser attiva una chiamata a `addV()`.

Consulta Combinazione di upsert e inserimenti per informazioni su come gestire le operazioni che possono emettere più traverser.

Upsert dei vertici

Il passaggio `mergeV()` è progettato specificamente per l'upsert dei vertici. Accetta come argomento un oggetto `Map` che rappresenta gli elementi di cui trovare una corrispondenza con i vertici esistenti nel grafo e, se non viene trovato un elemento, usa tale oggetto `Map` per creare un nuovo vertice. Il passaggio consente anche di modificare il comportamento in caso di creazione o corrispondenza, in cui è possibile applicare il modulatore `option()` con i token `Merge.onCreate` e `Merge.onMatch` per controllare i rispettivi comportamenti. Consulta la documentazione di TinkerPop riferimento per ulteriori informazioni su come utilizzare questo passaggio.

È possibile utilizzare un ID vertice per determinare se esiste un vertice specifico. Questo è l'approccio preferito, perché Neptune ottimizza gli upsert per casi d'uso altamente simultanei in base agli ID. Ad esempio, la seguente query crea un vertice con un determinato ID vertice se non esiste già o lo riutilizza se esiste:

```
g.mergeV([(T.id): 'v-1']).
    option(onCreate, [(T.label): 'PERSON', email: 'person-1@example.org', age: 21]).
    option(onMatch, [age: 22]).
  id()
```

Tenere presente che questa query termina con un passaggio `id()` Sebbene non sia strettamente necessario ai fini dell'upsert del vertice, un passaggio `id()` alla fine di una query di upsert assicura

che il server non serializzi tutte le proprietà del vertice sul client, riducendo il costo di blocco della query.

In alternativa, è possibile utilizzare una proprietà del vertice per identificare un vertice:

```
g.mergeV([email: 'person-1@example.org']).
    option(onCreate, [(T.label): 'PERSON', age: 21]).
    option(onMatch, [age: 22]).
  id()
```

Se possibile, utilizzare gli ID forniti dall'utente per creare i vertici e utilizzare questi ID per determinare se esiste un vertice durante un'operazione di upsert. Ciò consente a Neptune di ottimizzare gli upsert. Un upsert basato su ID può essere significativamente più efficiente di un upsert basato su proprietà quando le modifiche simultanee sono frequenti.

Concatenamento degli upsert dei vertici

È possibile concatenare gli upsert dei vertici per inserirli in un batch:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-1')
                        .property('email', 'person-1@example.org'))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-2')
                        .property('email', 'person-2@example.org'))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'v-3')
                        .property('email', 'person-3@example.org'))
 .id()
```

In alternativa, è possibile anche utilizzare questa sintassi `mergeV()`:

```
g.mergeV([(T.id): 'v-1', (T.label): 'PERSON', email: 'person-1@example.org']).
  mergeV([(T.id): 'v-2', (T.label): 'PERSON', email: 'person-2@example.org']).
  mergeV([(T.id): 'v-3', (T.label): 'PERSON', email: 'person-3@example.org'])
```

Tuttavia, poiché questo tipo di query include elementi nei criteri di ricerca che sono superflui rispetto alla ricerca di base con `id`, non è efficiente quanto la query precedente.

Upsert degli archi

Il passaggio `mergeE()` è progettato specificamente per l'upsert degli archi. Accetta come argomento un oggetto `Map` che rappresenta gli elementi di cui trovare una corrispondenza con gli archi esistenti nel grafo e, se non viene trovato un elemento, usa tale oggetto `Map` per creare un nuovo arco. Il passaggio consente anche di modificare il comportamento in caso di creazione o corrispondenza, in cui è possibile applicare il modulatore `option()` con i token `Merge.onCreate` e `Merge.onMatch` per controllare i rispettivi comportamenti. Consulta la documentazione di TinkerPop riferimento per ulteriori informazioni su come utilizzare questo passaggio.

È possibile utilizzare gli ID archi per eseguire l'upsert degli archi nello stesso modo in cui si esegue l'upsert dei vertici utilizzando gli ID vertice personalizzati. Anche in questo caso, si tratta dell'approccio preferito perché consente a Neptune di ottimizzare la query. Ad esempio, la seguente query crea un arco in base al relativo ID arco se non esiste già oppure lo riutilizza se esiste. La query utilizza anche gli ID dei vertici `Direction.from` e `Direction.to` se deve creare un nuovo arco:

```
g.mergeE([(T.id): 'e-1']).
    option(onCreate, [(from): 'v-1', (to): 'v-2', weight: 1.0]).
    option(onMatch, [weight: 0.5]).
  id()
```

Tenere presente che questa query termina con un passaggio `id()` Sebbene non sia strettamente necessario ai fini dell'upsert dell'arco, l'aggiunta di un passaggio `id()` alla fine di una query di upsert assicura che il server non serializzi tutte le proprietà dell'arco sul client, riducendo il costo di blocco della query.

Molte applicazioni utilizzano ID vertice personalizzati, ma lasciano che sia Neptune a generare gli ID arco. Se non si conosce l'ID di un arco, ma si conoscono gli ID vertice `from` e `to`, è possibile usare questo tipo di query per eseguire l'upsert di un arco:

```
g.mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  id()
```

Tutti i vertici a cui fa riferimento `mergeE()` devono esistere affinché il passaggio crei l'arco.

Concatenamento degli upsert degli archi

Come per gli upsert dei vertici, è semplice concatenare i passaggi `mergeE()` per le richieste batch:

```
g.mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  mergeE([(from): 'v-2', (to): 'v-3', (T.label): 'KNOWS']).
  mergeE([(from): 'v-3', (to): 'v-4', (T.label): 'KNOWS']).
  id()
```

Combinazione di upsert di vertici e archi

A volte si può desiderare di eseguire l'upsert sia dei vertici che degli archi che li collegano. È possibile combinare gli esempi batch illustrati qui. L'esempio seguente esegue l'upsert di 3 vertici e 2 archi:

```
g.mergeV([(id):'v-1']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org']).
  mergeV([(id):'v-2']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-2@example.org']).
  mergeV([(id):'v-3']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-3@example.org']).
  mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  mergeE([(from): 'v-2', (to): 'v-3', (T.label): 'KNOWS']).
 id()
```

Combinazione di upsert e inserimenti

A volte si può desiderare di eseguire l'upsert sia dei vertici che degli archi che li collegano. È possibile combinare gli esempi batch illustrati qui. L'esempio seguente esegue l'upsert di 3 vertici e 2 archi:

Gli upsert in genere procedono un elemento alla volta. Se ci si attiene ai modelli di upsert qui presentati, ogni operazione di upsert emette un singolo traverser, che fa sì che l'operazione successiva venga eseguita una sola volta.

Tuttavia, a volte si può voler combinare gli upsert con gli inserimenti. Questo può essere il caso, ad esempio, se si usano gli archi per rappresentare istanze di azioni o eventi. Una richiesta potrebbe utilizzare gli upsert per assicurarsi che esistano tutti i vertici necessari e quindi utilizzare gli inserimenti per aggiungere gli archi. Con richieste di questo tipo, occorre prestare attenzione al numero potenziale di traverser emessi da ciascuna operazione.

Considerare l'esempio seguente, che combina upsert e inserimenti per aggiungere archi che rappresentano gli eventi nel grafo:

```
// Fully optimized, but inserts too many edges
g.mergeV([(id):'v-1']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org']).
  mergeV([(id):'v-2']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-2@example.org']).
  mergeV([(id):'v-3']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-3@example.org']).
  mergeV([(T.id): 'c-1', (T.label): 'CITY', name: 'city-1']).
  V('p-1', 'p-2').
  addE('FOLLOWED').to(V('p-1')).
  V('p-1', 'p-2', 'p-3').
  addE('VISITED').to(V('c-1')).
  id()
```

La query deve inserire 5 archi: 2 archi FOLLOWED e 3 archi VISITED. Tuttavia, la query così com'è scritta inserisce 8 archi: 2 FOLLOWED e 6 VISITED. Il motivo di ciò è che l'operazione che inserisce i 2 archi FOLLOWED emette 2 traverser, facendo sì che la successiva operazione di inserimento, che inserisce 3 archi, venga eseguita due volte.

La soluzione consiste nell'aggiungere un passaggio `fold()` dopo ogni operazione che può potenzialmente emettere più di un traverser:

```
g.mergeV([(T.id): 'v-1', (T.label): 'PERSON', email: 'person-1@example.org']).
  mergeV([(T.id): 'v-2', (T.label): 'PERSON', email: 'person-2@example.org']).
  mergeV([(T.id): 'v-3', (T.label): 'PERSON', email: 'person-3@example.org']).
  mergeV([(T.id): 'c-1', (T.label): 'CITY', name: 'city-1']).
  V('p-1', 'p-2').
  addE('FOLLOWED').
    to(V('p-1')).
  fold().
  V('p-1', 'p-2', 'p-3').
  addE('VISITED').
    to(V('c-1')).
  id()
```

In questo caso è stato inserito un passaggio `fold()` dopo l'operazione che inserisce gli archi FOLLOWED. In questo modo si ottiene un singolo traverser, che fa sì che l'operazione successiva venga eseguita una sola volta.

Lo svantaggio di questo approccio è che la query ora non è completamente ottimizzata, perché `fold()` non è ottimizzato. Anche l'operazione di inserimento che segue `fold()` ora non sarà ottimizzata.

Se è necessario usare `fold()` per ridurre il numero di traverser per i passaggi successivi, provare a ordinare le operazioni in modo che quelle meno costose occupino la parte non ottimizzata della query.

Creazione di upsert Gremlin efficienti con **`fold()/coalesce()/unfold()`**

Un upsert (o inserimento condizionale) riutilizza un vertice o uno arco se già esiste oppure lo crea se non esiste. Upsert efficienti possono fare una differenza significativa nelle prestazioni delle query Gremlin.

Questa pagina mostra come utilizzare il modello Gremlin `fold()/coalesce()/unfold()` per creare upsert efficienti. Tuttavia, con il rilascio della TinkerPop versione 3.6.x introdotta in Neptune nella versione 1.2.1.0 del motore, nella maggior parte dei casi sono preferibili le nuove `mergeV()` fasi. `mergeE()` Il modello `fold()/coalesce()/unfold()` qui descritto può essere ancora utile in alcune situazioni complesse, ma in generale, se possibile, utilizzare `mergeV()` e `mergeE()`, come descritto in Creazione di upsert efficienti con i passaggi `mergeV()` e `mergeE()` di Gremlin.

Gli upsert consentono di scrivere operazioni di inserimento idempotenti: indipendentemente dal numero di volte in cui si esegue un'operazione di questo tipo, il risultato complessivo è lo stesso. Ciò è utile in scenari di scrittura altamente simultanei in cui modifiche simultanee alla stessa parte del grafo possono forzare il rollback di una o più transazioni con un'eccezione `ConcurrentModificationException`, rendendo quindi necessari un nuovo tentativo.

Ad esempio, la seguente query esegue l'upsert di un vertice cercando prima il vertice specificato nel set di dati e quindi raggruppando i risultati in un elenco. Nel primo attraversamento fornito al passaggio `coalesce()`, la query espande quindi questo elenco. Se l'elenco espanso non è vuoto, i risultati vengono emessi da `coalesce()`. Se, tuttavia, `unfold()` restituisce una raccolta vuota perché il vertice attualmente non esiste, `coalesce()` passa a valutare il secondo attraversamento che gli è stato fornito e in questo secondo attraversamento la query crea il vertice mancante.

```
g.V('v-1').fold()
          .coalesce(
             unfold(),
```

```
            addV('Person').property(id, 'v-1')
                          .property('email', 'person-1@example.org')
          )
```

Utilizzo di una forma ottimizzata di **`coalesce()`** per gli upsert

Neptune può ottimizzare l'idioma `fold().coalesce(unfold(), ...)` per apportare aggiornamenti con velocità di trasmissione effettiva elevata, ma questa ottimizzazione funziona solo se entrambe le parti di `coalesce()` restituiscono un vertice o un arco e nient'altro. Se si tenta di restituire qualcosa di diverso, ad esempio una proprietà, da qualsiasi parte di `coalesce()`, l'ottimizzazione di Neptune non si verifica. La query può avere esito positivo, ma non avrà le stesse prestazioni di una versione ottimizzata, in particolare su set di dati di grandi dimensioni.

Poiché le query upsert non ottimizzate aumentano i tempi di esecuzione e riducono la velocità di trasmissione effettiva, vale la pena utilizzare l'endpoint Gremlin `explain` per determinare se una query upsert è completamente ottimizzata. Quando si esaminano i piani `explain`, cercare le righe che iniziano con `+ not converted into Neptune steps` e `WARNING: >>`. Per esempio:

```
+ not converted into Neptune steps: [FoldStep, CoalesceStep([[UnfoldStep],
 [AddEdgeSte...
WARNING: >> FoldStep << is not supported natively yet
```

Questi avvisi consentono di identificare le parti di una query che ne impediscono l'ottimizzazione completa.

A volte non è possibile ottimizzare completamente una query. In queste situazioni è consigliabile provare a inserire i passaggi che non possono essere ottimizzati alla fine della query, in modo da consentire al motore di ottimizzare il maggior numero possibile di passaggi. Questa tecnica viene utilizzata in alcuni esempi di upsert in batch, in cui tutti gli upsert ottimizzati per un insieme di vertici o archi vengono eseguiti prima di applicare eventuali modifiche aggiuntive, potenzialmente non ottimizzate, agli stessi vertici o archi.

Upsert in batch per migliorare la velocità di trasmissione effettiva

Per scenari di scrittura con velocità di trasmissione effettiva elevata, è possibile concatenare i passaggi per eseguire l'upsert di vertici e archi in batch. Il batching riduce il sovraccarico transazionale dell'upsert di un gran numero di vertici e archi. È quindi possibile migliorare ulteriormente la velocità di trasmissione effettiva eseguendo l'upsert di richieste batch in parallelo utilizzando più client.

Come regola generale, è consigliabile eseguire l'upsert di circa 200 record per richiesta batch. Un record è una singola etichetta o proprietà di un vertice o un arco. Un vertice con una singola etichetta e 4 proprietà, ad esempio, crea 5 record. Un arco con un'etichetta e una singola proprietà crea 2 record. Se si vuole eseguire l'upsert di batch di vertici, ciascuno con una singola etichetta e 4 proprietà, è necessario iniziare con una dimensione di batch di 40, perché `200 / (1 + 4) = 40`.

È possibile sperimentare con la dimensione del batch. 200 record per batch sono un buon punto di partenza, ma la dimensione ideale del batch può essere maggiore o minore a seconda del carico di lavoro. Tenere presente, tuttavia, che Neptune può limitare il numero complessivo di passaggi Gremlin per richiesta. Questo limite non è documentato, ma per sicurezza, cercare di assicurarsi che le richieste non contengano più di 1.500 passaggi Gremlin. Neptune può rifiutare richieste batch di grandi dimensioni con più di 1.500 passaggi.

Per aumentare la velocità di trasmissione effettiva, è possibile eseguire l'upsert di batch in parallelo utilizzando più client (vedi Creazione di scritture multithread Gremlin efficienti). Il numero di client deve essere uguale al numero di thread di lavoro sull'istanza di scrittura Neptune, che in genere è 2 volte il numero di vCPU sul server. Ad esempio, un'istanza `r5.8xlarge` ha 32 vCPU e 64 thread di lavoro. Per scenari di scrittura con velocità di trasmissione effettiva elevata che utilizzano un'istanza `r5.8xlarge`, è necessario utilizzare 64 client che scrivono upsert in batch su Neptune in parallelo.

Ogni client deve inviare una richiesta batch e attendere il completamento della richiesta prima di inviarne un'altra. Sebbene più client vengano eseguiti in parallelo, ogni singolo client invia le richieste in modo seriale. Ciò garantisce che il server riceva un flusso costante di richieste che occupano tutti i thread di lavoro senza sovraccaricare la coda delle richieste lato server (vedi Dimensionamento delle istanze database in un cluster database Neptune).

Cercare di evitare passaggi che generano più traverser

Quando viene eseguito un passaggio Gremlin, accetta un traverser in entrata ed emette uno o più traverser in uscita. Il numero di traverser emessi da un passaggio determina il numero di volte in cui viene eseguito il passaggio successivo.

In genere, quando si eseguono operazioni in batch, si desidera che ogni operazione, ad esempio l'upsert del vertice A, venga eseguita una sola volta, in modo che la sequenza di operazioni sia la seguente: upsert del vertice A, quindi upsert del vertice B, quindi upsert del vertice C e così via. Finché un passaggio crea o modifica un solo elemento, emette un solo traverser e i passaggi che rappresentano l'operazione successiva vengono eseguiti una sola volta. Se, invece, un'operazione crea o modifica più di un elemento, emette più traverser, che a loro volta fanno sì che i passaggi

successivi vengano eseguiti più volte, una volta per ogni traverser emesso. Ciò può comportare l'esecuzione di operazioni aggiuntive non necessarie nel database e, in alcuni casi, la creazione di vertici, archi o valori di proprietà aggiuntivi indesiderati.

Un esempio di come le cose possano andare male è con una query come `g.V().addV()`. Questa semplice query aggiunge un vertice per ogni vertice trovato nel grafo, perché `V()` emette un traverser per ogni vertice del grafo e ognuno di questi traverser attiva una chiamata a `addV()`.

Consulta Combinazione di upsert e inserimenti per informazioni su come gestire le operazioni che possono emettere più traverser.

Upsert dei vertici

È possibile utilizzare un ID vertice per determinare se esiste un vertice corrispondente. Questo è l'approccio preferito, perché Neptune ottimizza gli upsert per casi d'uso altamente simultanei in base agli ID. Ad esempio, la seguente query crea un vertice con un determinato ID vertice se non esiste già o lo riutilizza se esiste:

```
g.V('v-1')
 .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-1')
                          .property('email', 'person-1@example.org'))
  .id()
```

Tenere presente che questa query termina con un passaggio `id()` Sebbene non sia strettamente necessario ai fini dell'upsert del vertice, l'aggiunta di un passaggio `id()` alla fine di una query di upsert assicura che il server non serializzi tutte le proprietà del vertice sul client, riducendo il costo di blocco della query.

In alternativa, è possibile utilizzare una proprietà del vertice per determinare se il vertice esiste:

```
g.V()
 .hasLabel('Person')
 .has('email', 'person-1@example.org')
 .fold()
 .coalesce(unfold(),
           addV('Person').property('email', 'person-1@example.org'))
 .id()
```

Se possibile, utilizzare gli ID forniti dall'utente per creare i vertici e utilizzare questi ID per determinare se esiste un vertice durante un'operazione di upsert. Ciò consente a Neptune di ottimizzare gli upsert in base agli ID. Un upsert basato su ID può essere significativamente più efficiente di un upsert basato su proprietà in scenari di modifica altamente simultanei.

Concatenamento degli upsert dei vertici

È possibile concatenare gli upsert dei vertici per inserirli in un batch:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org'))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org'))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-3')
                         .property('email', 'person-3@example.org'))
 .id()
```

Upsert degli archi

È possibile utilizzare gli ID archi per eseguire l'upsert degli archi nello stesso modo in cui si esegue l'upsert dei vertici utilizzando gli ID vertice personalizzati. Anche in questo caso, si tratta dell'approccio preferito perché consente a Neptune di ottimizzare la query. Ad esempio, la seguente query crea un arco in base al relativo ID arco se non esiste già oppure lo riutilizza se esiste. La query utilizza anche gli ID dei vertici `from` e `to` se deve creare un nuovo arco.

```
g.E('e-1')
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2'))
                        .property(id, 'e-1'))
 .id()
```

Molte applicazioni utilizzano ID vertice personalizzati, ma lasciano che sia Neptune a generare gli ID arco. Se non si conosce l'ID di un arco, ma si conoscono gli ID vertice `from` e `to`, è possibile usare questa formulazione per eseguire l'upsert di un arco:

```
g.V('v-1')
 .outE('KNOWS')
 .where(inV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2')))
 .id()
```

Notare che il passaggio del vertice nella clausola `where()` deve essere `inV()` (o `outV()` se è stato usato `inE()` per trovare l'arco), non `otherV()`. Non utilizzare `otherV()` in questo caso, altrimenti la query non verrà ottimizzata e le prestazioni ne risentiranno. Ad esempio, Neptune non ottimizzerà la seguente query:

```
// Unoptimized upsert, because of otherV()
g.V('v-1')
 .outE('KNOWS')
 .where(otherV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2')))
 .id()
```

Se non si conoscono gli ID degli archi o dei vertici, è possibile eseguire l'upsert utilizzando le proprietà dei vertici:

```
g.V()
 .hasLabel('Person')
 .has('name', 'person-1')
 .outE('LIVES_IN')
 .where(inV().hasLabel('City').has('name', 'city-1'))
 .fold()
 .coalesce(unfold(),
           addE('LIVES_IN').from(V().hasLabel('Person')
                                    .has('name', 'person-1'))
                           .to(V().hasLabel('City')
```

```
                                    .has('name', 'city-1')))
  .id()
```

Come per l'upsert dei vertici, è preferibile utilizzare gli upsert degli archi basati su ID utilizzando un ID arco o gli ID vertice `from` e `to`, piuttosto che gli upsert basati sulle proprietà, in modo che Neptune possa ottimizzare completamente l'upsert.

Verifica dell'esistenza dei vertici **`from`** e **`to`**

Notare la costruzione dei passaggi che creano un nuovo arco: `addE().from().to()`. Questa costruzione assicura che la query verifichi l'esistenza sia del vertice `from` che del vertice `to`. Se uno dei due non esiste, la query restituisce un errore come segue:

```
{
  "detailedMessage": "Encountered a traverser that does not map to a value for child...
  "code": "IllegalArgumentException",
  "requestId": "..."
}
```

Se è possibile che il vertice `from` o il vertice `to` non esista, è consigliabile provare a eseguirne l'upsert prima di eseguire l'upsert dell'arco tra di loro. Per informazioni, consulta Combinazione di upsert di vertici e archi.

Esiste una costruzione alternativa per la creazione di un arco che non è consigliabile usare: `V().addE().to()`. Aggiunge uno arco solo se esiste il vertice `from`. Se il vertice `to` non esiste, la query genera un errore, come descritto in precedenza, ma se il vertice `from` non esiste, l'inserimento di un arco non riesce ma non viene generato alcun errore. Ad esempio, il seguente upsert viene completato senza eseguire l'upsert di un arco se il vertice `from` non esiste:

```
// Will not insert edge if from vertex does not exist
g.V('v-1')
  .outE('KNOWS')
  .where(inV().hasId('v-2'))
  .fold()
  .coalesce(unfold(),
            V('v-1').addE('KNOWS')
                    .to(V('v-2')))
  .id()
```

Concatenamento degli upsert degli archi

Se si desidera concatenare gli upsert degli archi per creare una richiesta batch, è necessario iniziare ogni upsert con una ricerca dei vertici, anche se si conoscono già gli ID degli archi.

Se si conoscono già gli ID degli archi di cui si vuole eseguire l'upsert e gli ID dei vertici `from` e `to`, è possibile usare questa formulazione:

```
g.V('v-1')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
           V('v-1').addE('KNOWS')
                   .to(V('v-2'))
                   .property(id, 'e-1'))
 .V('v-3')
 .outE('KNOWS')
 .hasId('e-2').fold()
 .coalesce(unfold(),
           V('v-3').addE('KNOWS')
                   .to(V('v-4'))
                   .property(id, 'e-2'))
 .V('v-5')
 .outE('KNOWS')
 .hasId('e-3')
 .fold()
 .coalesce(unfold(),
           V('v-5').addE('KNOWS')
                   .to(V('v-6'))
                   .property(id, 'e-3'))
 .id()
```

Forse lo scenario di upsert degli archi in batch più comune è quello in cui si conoscono gli ID dei vertici `from` e `to` ma non si conoscono gli ID degli archi di cui si desidera eseguire l'upsert. In tal caso, utilizzare la seguente formulazione:

```
g.V('v-1')
 .outE('KNOWS')
 .where(inV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
```

```
          V('v-1').addE('KNOWS')
                  .to(V('v-2')))

 .V('v-3')
 .outE('KNOWS')
 .where(inV().hasId('v-4'))
 .fold()
 .coalesce(unfold(),
          V('v-3').addE('KNOWS')
                  .to(V('v-4')))
 .V('v-5')
 .outE('KNOWS')
 .where(inV().hasId('v-6'))
 .fold()
 .coalesce(unfold(),
          V('v-5').addE('KNOWS').to(V('v-6')))
 .id()
```

Se si conoscono gli ID degli archi di cui si vuole eseguire l'upsert, ma non si conoscono gli ID dei vertici `from` e `to` (caso insolito), è possibile usare questa formulazione:

```
g.V()
 .hasLabel('Person')
 .has('email', 'person-1@example.org')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
          V().hasLabel('Person')
             .has('email', 'person-1@example.org')
             .addE('KNOWS')
             .to(V().hasLabel('Person')
                    .has('email', 'person-2@example.org'))
                    .property(id, 'e-1'))
 .V()
 .hasLabel('Person')
 .has('email', 'person-3@example.org')
 .outE('KNOWS')
 .hasId('e-2')
 .fold()
 .coalesce(unfold(),
          V().hasLabel('Person')
             .has('email', 'person-3@example.org')
```

```
              .addE('KNOWS')
              .to(V().hasLabel('Person')
                     .has('email', 'person-4@example.org'))
              .property(id, 'e-2'))
 .V()
 .hasLabel('Person')
 .has('email', 'person-5@example.org')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
           V().hasLabel('Person')
              .has('email', 'person-5@example.org')
              .addE('KNOWS')
              .to(V().hasLabel('Person')
                     .has('email', 'person-6@example.org'))
                     .property(id, 'e-3'))
 .id()
```

Combinazione di upsert di vertici e archi

A volte si può desiderare di eseguire l'upsert sia dei vertici che degli archi che li collegano. È possibile combinare gli esempi batch illustrati qui. L'esempio seguente esegue l'upsert di 3 vertici e 2 archi:

```
g.V('p-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-1')
                         .property('email', 'person-1@example.org'))
 .V('p-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-2')
                         .property('name', 'person-2@example.org'))
 .V('c-1')
 .fold()
 .coalesce(unfold(),
           addV('City').property(id, 'c-1')
                       .property('name', 'city-1'))
 .V('p-1')
 .outE('LIVES_IN')
 .where(inV().hasId('c-1'))
 .fold()
```

```
 .coalesce(unfold(),
           V('p-1').addE('LIVES_IN')
                   .to(V('c-1')))
 .V('p-2')
 .outE('LIVES_IN')
 .where(inV().hasId('c-1'))
 .fold()
 .coalesce(unfold(),
           V('p-2').addE('LIVES_IN')
                   .to(V('c-1')))
 .id()
```

Combinazione di upsert e inserimenti

A volte si può desiderare di eseguire l'upsert sia dei vertici che degli archi che li collegano. È possibile combinare gli esempi batch illustrati qui. L'esempio seguente esegue l'upsert di 3 vertici e 2 archi:

Gli upsert in genere procedono un elemento alla volta. Se ci si attiene ai modelli di upsert qui presentati, ogni operazione di upsert emette un singolo traverser, che fa sì che l'operazione successiva venga eseguita una sola volta.

Tuttavia, a volte si può voler combinare gli upsert con gli inserimenti. Questo può essere il caso, ad esempio, se si usano gli archi per rappresentare istanze di azioni o eventi. Una richiesta potrebbe utilizzare gli upsert per assicurarsi che esistano tutti i vertici necessari e quindi utilizzare gli inserimenti per aggiungere gli archi. Con richieste di questo tipo, occorre prestare attenzione al numero potenziale di traverser emessi da ciascuna operazione.

Considerare l'esempio seguente, che combina upsert e inserimenti per aggiungere archi che rappresentano gli eventi nel grafo:

```
// Fully optimized, but inserts too many edges
g.V('p-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-1')
                         .property('email', 'person-1@example.org'))
 .V('p-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-2')
                         .property('name', 'person-2@example.org'))
```

```
 .V('p-3')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'p-3')
                        .property('name', 'person-3@example.org'))
 .V('c-1')
 .fold()
 .coalesce(unfold(),
          addV('City').property(id, 'c-1')
                      .property('name', 'city-1'))
 .V('p-1', 'p-2')
 .addE('FOLLOWED')
 .to(V('p-1'))
 .V('p-1', 'p-2', 'p-3')
 .addE('VISITED')
 .to(V('c-1'))
 .id()
```

La query deve inserire 5 archi: 2 archi FOLLOWED e 3 archi VISITED. Tuttavia, la query così com'è scritta inserisce 8 archi: 2 FOLLOWED e 6 VISITED. Il motivo di ciò è che l'operazione che inserisce i 2 archi FOLLOWED emette 2 traverser, facendo sì che la successiva operazione di inserimento, che inserisce 3 archi, venga eseguita due volte.

La soluzione consiste nell'aggiungere un passaggio `fold()` dopo ogni operazione che può potenzialmente emettere più di un traverser:

```
g.V('p-1')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'p-1')
                        .property('email', 'person-1@example.org'))
 .V('p-2')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'p-2').
                        .property('name', 'person-2@example.org'))
 .V('p-3')
 .fold()
 .coalesce(unfold(),
          addV('Person').property(id, 'p-3').
                        .property('name', 'person-3@example.org'))
 .V('c-1')
 .fold().
```

```
 .coalesce(unfold(),
           addV('City').property(id, 'c-1').
                       .property('name', 'city-1'))
 .V('p-1', 'p-2')
 .addE('FOLLOWED')
 .to(V('p-1'))
 .fold()
 .V('p-1', 'p-2', 'p-3')
 .addE('VISITED')
 .to(V('c-1')).
 .id()
```

In questo caso è stato inserito un passaggio `fold()` dopo l'operazione che inserisce gli archi FOLLOWED. In questo modo si ottiene un singolo traverser, che fa sì che l'operazione successiva venga eseguita una sola volta.

Lo svantaggio di questo approccio è che la query ora non è completamente ottimizzata, perché `fold()` non è ottimizzato. L'operazione di inserimento che segue `fold()` ora non sarà ottimizzata.

Se è necessario usare `fold()` per ridurre il numero di traverser per i passaggi successivi, provare a ordinare le operazioni in modo che quelle meno costose occupino la parte non ottimizzata della query.

Upsert che modificano vertici e archi esistenti

A volte si desidera creare un vertice o un arco se non esiste e quindi aggiungere o aggiornare una proprietà, indipendentemente dal fatto che si tratti di un vertice o di un arco nuovo o esistente.

Per aggiungere o modificare una proprietà, utilizzare il passaggio `property()`. Utilizzare questo passaggio all'esterno del passaggio `coalesce()`. Se si prova a modificare la proprietà di un vertice o di un arco esistente all'interno del passaggio `coalesce()`, la query potrebbe non essere ottimizzata dal motore di query Neptune.

La seguente query aggiunge o aggiorna la proprietà counter su ogni vertice di cui è stato eseguito l'upsert. Ogni passaggio `property()` ha una cardinalità singola per garantire che i nuovi valori sostituiscano quelli esistenti, anziché essere aggiunti a un insieme di valori esistenti.

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org'))
```

```
  .property(single, 'counter', 1)
  .V('v-2')
  .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-2')
                          .property('email', 'person-2@example.org'))
  .property(single, 'counter', 2)
  .V('v-3')
  .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-3')
                          .property('email', 'person-3@example.org'))
  .property(single, 'counter', 3)
  .id()
```

Se si dispone di un valore di proprietà, ad esempio il valore di timestamp `lastUpdated`, che si applica a tutti gli elementi di cui è stato eseguito l'upsert, è possibile aggiungerlo o aggiornarlo alla fine della query:

```
g.V('v-1')
  .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-1')
                          .property('email', 'person-1@example.org'))
  .V('v-2').
  .fold().
  .coalesce(unfold(),
            addV('Person').property(id, 'v-2')
                          .property('email', 'person-2@example.org'))
  .V('v-3')
  .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-3')
                          .property('email', 'person-3@example.org'))
  .V('v-1', 'v-2', 'v-3')
  .property(single, 'lastUpdated', datetime('2020-02-08'))
  .id()
```

Se esistono condizioni aggiuntive che determinano se un vertice o un arco debba essere ulteriormente modificato, è possibile utilizzare un passaggio `has()` per filtrare gli elementi a cui verrà applicata una modifica. L'esempio seguente utilizza un passaggio `has()` per filtrare i vertici di cui è

stato eseguito l'upsert in base al valore della relativa proprietà `version`. La query quindi aggiorna a 3 il valore `version` di qualsiasi vertice il cui valore `version` sia inferiore a 3:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org')
                         .property('version', 3))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org')
                         .property('version', 3))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-3')
                         .property('email', 'person-3@example.org')
                         .property('version', 3))
 .V('v-1', 'v-2', 'v-3')
 .has('version', lt(3))
 .property(single, 'version', 3)
 .id()
```

Analisi dell'esecuzione di query Neptune tramite la funzionalità Gremlin **`explain`**

Amazon Neptune ha aggiunto una funzionalità Gremlin denominata explain. Questa funzionalità è uno strumento self-service che consente di comprendere l'approccio di esecuzione adottato dal motore Neptune. È possibile richiamarla aggiungendo un parametro `explain` a una chiamata HTTP che invia una query Gremlin.

La funzione `explain` fornisce informazioni sulla struttura logica dei piani di esecuzione di query. Puoi utilizzare queste informazioni per identificare potenziali problemi di valutazione ed esecuzione, oltre che per ottimizzare la query, come spiegato in Ottimizzazione di query Gremlin. Puoi quindi utilizzare gli hint di query per migliorare i piani di esecuzione delle query.

> **Note**
>
> Questa funzionalità è disponibile a partire dal Rilascio 1.0.1.0.200463.0 (15/10/2019).

Argomenti

Introduzione al funzionamento delle query Gremlin in Neptune

Per sfruttare appieno i report `explain` e `profile` di Gremlin in Amazon Neptune, è utile comprendere alcune informazioni generali sulle query Gremlin.

Argomenti

Dichiarazioni Gremlin in Neptune

I dati del grafo delle proprietà in Amazon Neptune sono costituiti da dichiarazioni a quattro posizioni (quadruple). Ognuna di queste dichiarazioni rappresenta una singola unità atomica dei dati del grafo di proprietà. Per ulteriori informazioni, consulta Modello di dati a grafo di Neptune. Analogamente al modello di dati Resource Description Framework (RDF), queste quattro posizioni sono le seguenti:

- `subject (S)`
- `predicate (P)`
- `object (O)`
- `graph (G)`

Ogni dichiarazione è un'asserzione su una o più risorse. Ad esempio, una dichiarazione può asserire l'esistenza di una relazione tra due risorse o può collegare una proprietà (coppia chiave-valore) ad alcune risorse.

Il predicato può essere considerato come il verbo della dichiarazione, che descrive il tipo di relazione o la proprietà. L'oggetto è la destinazione della relazione o il valore della proprietà. La posizione del grafo è opzionale e può essere utilizzata in diversi modi. Per i dati del grafo delle proprietà (PG) Neptune, questa è inutilizzata (grafico null) o è utilizzata per rappresentare l'identificatore di un arco. Un set di dichiarazioni con identificatori di risorsa condivisi crea un grafo.

Esistono tre classi di dichiarazioni nel modello di dati del grafo delle proprietà Neptune:

Argomenti

Dichiarazioni dell'etichetta del vertice Gremlin

Le dichiarazioni dell'etichetta del vertice in Neptune hanno due scopi:

- Tracciare le etichette per un vertice.
- La presenza di almeno una di queste dichiarazioni implica di per sé l'esistenza di un determinato vertice nel grafo.

Il soggetto di queste dichiarazioni è un identificatore del vertice e l'oggetto è un'etichetta, entrambi specificati dall'utente. È possibile utilizzare uno predicato fisso speciale per queste dichiarazioni, visualizzato come `<~label>` e un identificatore del grafo predefinito (il grafo null), visualizzato come `<~>`.

Considera, ad esempio, l'attraversamento `addV` seguente:

```
g.addV("Person").property(id, "v1")
```

Questo attraversamento determina l'aggiunta della seguente dichiarazione al grafo.

```
StatementEvent[Added(<v1> <~label> <Person> <~>) .]
```

Dichiarazioni di edge Gremlin

Una dichiarazione di arco Gremlin implica di per sé l'esistenza di un arco tra due vertici in un grafo in Neptune. Il soggetto (S) di una dichiarazione di edge è il vertice `from` di origine. Il predicato (P) è un'etichetta di edge fornita dall'utente. L'oggetto (O) è il vertice `to` di destinazione. Il grafico (G) è un identificatore di edge fornito dall'utente.

Considera, ad esempio, l'attraversamento `addE` seguente:

```
g.addE("knows").from(V("v1")).to(V("v2")).property(id, "e1")
```

L'attraversamento comporta l'aggiunta della seguente dichiarazione al grafico.

```
StatementEvent[Added(<v1> <knows> <v2> <e1>) .]
```

Dichiarazioni di proprietà Gremlin

Una dichiarazione di proprietà Gremlin in Neptune afferma un singolo valore di proprietà per un vertice o un arco. Il soggetto è un identificatore di vertice o edge fornito dall'utente. Il predicato è il nome della proprietà (chiave) e l'oggetto è il singolo valore della proprietà. Il grafo (G) è di nuovo l'identificatore del grafo predefinito, il grafo null, visualizzato come <~>.

Analizza l'esempio seguente.

```
g.V("v1").property("name", "John")
```

Questa dichiarazione genera quanto segue.

```
StatementEvent[Added(<v1> <name> "John" <~>) .]
```

Le istruzioni di proprietà differiscono dalle altre in quanto il relativo oggetto è un valore primitivo (`string`, `date`, `byte`, `short`, `int`, `long`, `float` o `double`). Il loro oggetto non è un identificatore di risorsa che può essere utilizzato come il soggetto di un'altra asserzione.

Per le proprietà multiple, ogni singolo valore di proprietà nel set riceve la propria dichiarazione.

```
g.V("v1").property(set, "phone", "956-424-2563").property(set, "phone", "956-354-3692
 (tel:9563543692)")
```

I risultati sono illustrati di seguito.

```
StatementEvent[Added(<v1> <phone> "956-424-2563" <~>) .]
StatementEvent[Added(<v1> <phone> "956-354-3692" <~>) .]
```

Come Neptune elabora le query Gremlin mediante gli indici di dichiarazione

L'accesso alle dichiarazioni in Amazon Neptune avviene tramite tre indici di dichiarazioni, come descritto in In che modo le dichiarazioni vengono indicizzate in Neptune. Neptune estrae un modello di dichiarazione da una query Gremlin in cui alcune posizioni sono note mentre le altre vengono individuate tramite la ricerca nell'indice.

Neptune presuppone che le dimensioni dello schema del grafo delle proprietà non siano grandi. Ciò significa che il numero di etichette degli archi e di nomi di proprietà distinti è piuttosto basso, determinando un basso numero totale di predicati distinti. Neptune tiene traccia dei predicati distinti in un indice separato. Utilizza questa cache di predicati per eseguire una scansione dell'unione di `{ all P x POGS }` anziché utilizzare un indice OSPG. La possibilità di evitare un indice OSPG di attraversamento inverso permette di risparmiare spazio di storage e throughput del caricamento.

L'API Explain/Profile di Gremlin in Neptune consente di ottenere il conteggio dei predicati nel grafo. È possibile quindi determinare se l'applicazione invalida il presupposto di Neptune che le dimensioni dello schema del grafo delle proprietà siano ridotte.

I seguenti esempi illustrano come Neptune utilizza gli indici per elaborare le query Gremlin.

Domanda: quali sono le etichette dei vertici **v1**?

```
Gremlin code:      g.V('v1').label()
Pattern:           (<v1>, <~label>, ?, ?)
Known positions:   SP
Lookup positions:  OG
Index:             SPOG
Key range:         <v1>:<~label>:*
```

Domanda: quali sono gli edge "noti" dei vertici **v1**?

```
Gremlin code:      g.V('v1').out('knows')
Pattern:           (<v1>, <knows>, ?, ?)
Known positions:   SP
Lookup positions:  OG
Index:             SPOG
Key range:         <v1>:<knows>:*
```

Domanda: quali vertici hanno un'etichetta di vertice **`Person`**?

```
Gremlin code:      g.V().hasLabel('Person')
Pattern:           (?, <~label>, <Person>, <~>)
Known positions:   POG
Lookup positions:  S
Index:             POGS
Key range:         <~label>:<Person>:<~>:*
```

Domanda: quali sono i vertici da/a di un determinato edge **`e1`**?

```
Gremlin code:      g.E('e1').bothV()
Pattern:           (?, ?, ?, <e1>)
Known positions:   G
Lookup positions:  SPO
Index:             GPSO
Key range:         <e1>:*
```

Un indice di dichiarazione non presente in Neptune è un indice OSPG di attraversamento inverso. Questo indice può essere utilizzato per raccogliere tutti gli edge in entrata in tutte le etichette edge, come nell'esempio seguente.

Domanda: quali sono i vertici **v1** adiacenti in entrata?

```
Gremlin code:      g.V('v1').in()
Pattern:           (?, ?, <v1>, ?)
Known positions:   O
Lookup positions:  SPG
Index:             OSGP  // <-- Index does not exist
```

Come vengono elaborate le query Gremlin in Neptune

In Amazon Neptune, attraversamenti più complessi possono essere rappresentati da una serie di modelli che creano una relazione basata sulla definizione di variabili denominate che possono essere condivise tra modelli per creare join. Questo viene mostrato nell'esempio seguente.

Domanda: cos'è il quartiere a due hop del vertice **v1**?

```
Gremlin code:      g.V(‘v1’).out('knows').out('knows').path()
Pattern:           (?1=<v1>, <knows>, ?2, ?) X Pattern(?2, <knows>, ?3, ?)

The pattern produces a three-column relation (?1, ?2, ?3) like this:
```

```
                    ?1      ?2      ?3
                    ================
                    v1      v2      v3
                    v1      v2      v4
                    v1      v5      v6
```

Condividendo la variabile `?2` tra i due modelli (nella posizione O del primo modello e nella posizione S del secondo modello), si crea un join dai primi quartieri hop ai secondi quartieri hop. Ogni soluzione Neptune ha associazioni per le tre variabili denominate, che possono essere utilizzate per ricreare TinkerPopun Traverser (comprese le informazioni sul percorso).

Il primo passo nell'elaborazione delle query di Gremlin consiste nell'analizzare la query in un oggetto Traversal, composto da una TinkerPop serie di passaggi. TinkerPop Questi passaggi, che fanno parte del TinkerPop progetto open source Apache, sono sia gli operatori logici che fisici che compongono un attraversamento Gremlin nell'implementazione di riferimento. Entrambi vengono utilizzati per rappresentare il modello della query. Si tratta di operatori eseguibili che possono generare soluzioni in base alla semantica dell'operatore che rappresentano. Ad esempio, `.V()` è sia rappresentato che eseguito da. TinkerPop GraphStep

Poiché questi off-the-shelf TinkerPop passaggi sono eseguibili, un TinkerPop Traversal di questo tipo può eseguire qualsiasi interrogazione Gremlin e produrre la risposta corretta. Tuttavia, se eseguiti su un grafico di grandi dimensioni, TinkerPop i passaggi a volte possono essere molto inefficienti e lenti. Anziché usarle, Neptune cerca di convertire l'attraversamento in un formato dichiarativo composto da gruppi di modelli, come descritto in precedenza.

Neptune non supporta attualmente tutti gli operatori Gremlin (passaggi) nel motore di query nativo. Pertanto, tenta di comprimere il maggior numero di fasi possibili in una singola `NeptuneGraphQueryStep`, che contiene il piano di query logico dichiarativo per tutte le fasi che sono state convertite. Idealmente, tutte le fasi vengono convertite. Ma quando viene rilevato un passaggio che non può essere convertito, Neptune interrompe l'esecuzione nativa e rinvia tutta l'esecuzione delle query da quel momento in avanti ai passaggi. TinkerPop Non tenta di intrufolarsi nell'esecuzione nativa.

Dopo che i passaggi vengono tradotte in un piano di query logico, Neptune esegue una serie di ottimizzatori di query che riscrivono il piano di query in base all'analisi statica e alle cardinalità stimate. Questi ottimizzatori eseguono operazioni come riordinare operatori in base a conteggi di intervalli, eliminare operatori superflui o ridondanti, riordinare i filtri, eseguire il push di operatori in gruppi diversi e così via.

Dopo aver prodotto un piano di query ottimizzato, Neptune crea una pipeline di operatori fisici che eseguono il lavoro di esecuzione della query. Ciò include la lettura dei dati dagli indici di dichiarazione, l'esecuzione di join di vari tipi, il filtraggio, l'ordinamento e così via. La pipeline produce un flusso di soluzioni che viene poi riconvertito in un flusso di oggetti Traverser. TinkerPop

Serializzazione dei risultati delle query

Amazon Neptune attualmente si affida ai serializzatori TinkerPop dei messaggi di risposta per convertire i risultati delle query TinkerPop (Traversers) in dati serializzati da inviare via cavo al client. Questi formati di serializzazione tendono a essere piuttosto dettagliati.

Ad esempio, per serializzare il risultato di una query del vertice quale `g.V().limit(1)`, il motore di query Neptune deve eseguire una singola ricerca per produrre il risultato della query. Tuttavia, il serializzatore `GraphSON` eseguirà un numero elevato di ricerche aggiuntive per creare un pacchetto del vertice nel formato di serializzazione. Deve eseguire una ricerca per ottenere l'etichetta, una per ottenere le chiavi di proprietà e una ricerca per chiave di proprietà per il vertice per ottenere tutti i valori per ogni chiave.

Alcuni dei formati di serializzazione sono più efficienti, ma tutti richiedono ricerche aggiuntive. Inoltre, i TinkerPop serializzatori non cercano di evitare ricerche duplicate, spesso con il risultato che molte ricerche vengono ripetute inutilmente.

Questo rende molto importante scrivere le query in modo da richiedere specificamente solo le informazioni necessarie. Ad esempio, `g.V().limit(1).id()` restituisce solo l'ID del vertice ed elimina tutte le ricerche di serializzatore aggiuntive. API Gremlin `profile` in Neptune consente di vedere quante chiamate di ricerca vengono effettuate durante l'esecuzione della query e durante la serializzazione.

Utilizzo dell'API Gremlin **`explain`** in Neptune

L'API `explain` di Gremlin in Amazon Neptune restituisce il piano di query che verrebbe eseguito se fosse stata eseguita una determinata query. Poiché l'API non esegue effettivamente la query, il piano viene restituito quasi istantaneamente.

Si differenzia dal passaggio TinkerPop .explain () per poter riportare informazioni specifiche per il motore Neptune.

Informazioni contenute in un report Gremlin **`explain`**

Il report `explain` contiene le seguenti informazioni:

- La stringa di query che è stata richiesta.
- L'attraversamento originale. Questo è l'oggetto TinkerPop Traversal prodotto analizzando la stringa di query in passaggi. TinkerPop È equivalente alla query originale prodotta eseguendo la query `.explain()` su. TinkerPop TinkerGraph
- L'attraversamento convertito. Questo è il Neptune Traversal prodotto convertendo il Traversal nella rappresentazione del piano di interrogazione TinkerPop logica di Neptune. In molti casi l'intero TinkerPop attraversamento viene convertito in due passaggi di Neptune: uno che esegue l'intera query () `NeptuneGraphQueryStep` e uno che converte l'output del motore di query Neptune in Traversers (). TinkerPop `NeptuneTraverserConverterStep`
- L'attraversamento ottimizzato. Questa è la versione ottimizzata del piano di query Neptune dopo che è stato eseguito tramite una serie di ottimizzatori di riduzione del lavoro statici che riscrivono la query in base all'analisi statica e alle cardinalità stimate. Questi ottimizzatori eseguono operazioni come riordinare operatori in base a conteggi di intervalli, eliminare operatori superflui o ridondanti, riordinare i filtri, eseguire il push di operatori in gruppi diversi e così via.
- Il conteggio di predicati. A causa della strategia di indicizzazione Neptune descritta in precedenza, avere un numero elevato di predicati diversi può causare problemi di prestazioni. Ciò vale soprattutto per query che utilizzano operatori di attraversamento inverso senza etichetta edge (`.in` o `.both`). Se vengono utilizzati tali operatori e il conteggio di predicati è sufficientemente elevato, il report `explain` visualizza un messaggio di avviso.
- Informazioni su DFE. Quando il motore alternativo DFE è abilitato, nell'attraversamento ottimizzato possono apparire i seguenti componenti di attraversamento:
  - **`DFEStep`**: passaggio DFE ottimizzato per Neptune nell'attraversamento che contiene un oggetto `DFENode` figlio. `DFEStep` rappresenta la parte del piano di query che viene eseguita nel motore DFE.
  - **`DFENode`**: contiene la rappresentazione intermedia come uno o più oggetti `DFEJoinGroupNodes` figlio.
  - **`DFEJoinGroupNode`**: rappresenta l'unione di uno o più elementi `DFENode` o `DFEJoinGroupNode`.
  - **`NeptuneInterleavingStep`**: passaggio DFE ottimizzato per Neptune nell'attraversamento che contiene un oggetto `DFEStep` figlio.

    Contiene anche un elemento `stepInfo` che contiene informazioni sull'attraversamento, come l'elemento di frontiera, gli elementi di percorso utilizzati e così via. Queste informazioni vengono utilizzate per elaborare l'oggetto `DFEStep` figlio.

Un modo semplice per scoprire se la query viene valutata dal motore DFE consiste nel verificare se l'output `explain` contiene un oggetto `DFEStep`. Qualsiasi parte dell'attraversamento che non fa parte di non verrà eseguita da DFE e `DFEStep` verrà eseguita dal motore. TinkerPop

Vedi Esempio con DFE abilitato per un report di esempio.

Sintassi di Gremlin **`explain`**

La sintassi dell'API `explain` è identica a quella dell'API HTTP per la query, tranne per il fatto che utilizza `/gremlin/explain` come endpoint anziché `/gremlin`, come nell'esempio seguente.

```
curl -X POST https://your-neptune-endpoint:port/gremlin/explain -d
 '{"gremlin":"g.V().limit(1)"}'
```

La query precedente produrrebbe il seguente output.

```
********************************************************
                Neptune Gremlin Explain
********************************************************

Query String
============
g.V().limit(1)

Original Traversal
==================
[GraphStep(vertex,[]), RangeGlobalStep(0,1)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
        }, finishers=[limit(1)], annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
```

```
==================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, finishers=[limit(1)], annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 18
```

TinkerPop Passi non convertiti

Idealmente, tutte le TinkerPop fasi di un attraversamento hanno una copertura dell'operatore Neptune nativa. In caso contrario, Neptune ricorre all'esecuzione a fasi per eventuali lacune TinkerPop nella copertura degli operatori. Se un attraversamento utilizza un passaggio per il quale Neptune non dispone ancora di copertura nativa, il report `explain` visualizza un avviso che mostra dove si è verificata la lacuna.

Quando viene rilevato un passaggio senza un corrispondente operatore Neptune nativo, l'intero attraversamento da quel punto in poi viene eseguito TinkerPop utilizzando i passaggi, anche se i passaggi successivi hanno operatori Neptune nativi.

L'eccezione a questo è quando viene richiamata la ricerca full-text Neptune. NeptuneSearchStep Implementa passaggi senza equivalenti nativi come passaggi di ricerca a testo completo.

Esempio di output di **`explain`** in cui tutti i passaggi di una query dispongono di equivalenti nativi

Di seguito è riportato un esempio di report di `explain` per una query in cui tutti i passaggi dispongono di equivalenti nativi:

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============
g.V().out()
```

```
Original Traversal
==================
[GraphStep(vertex,[]), VertexStep(OUT,vertex)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 18
```

Esempio in cui alcune fasi in una query non dispongono di equivalenti nativi

Neptune gestisce `GraphStep` e `VertexStep` in modo nativo, ma se si introduce `FoldStep` e `UnfoldStep`, l'output risultante di `explain` è diverso:

```
************************************************************
                Neptune Gremlin Explain
```

```
*******************************************************

Query String
============
g.V().fold().unfold().out()

Original Traversal
==================
[GraphStep(vertex,[]), FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep,
    NeptuneMemoryTrackerStep
]
+ not converted into Neptune steps: [FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

WARNING: >> FoldStep << is not supported natively yet
```

In questo caso, `FoldStep` interrompe l'esecuzione nativa. Tuttavia, anche la `VertexStep` successiva non viene più gestita in modo nativo perché viene visualizzato a valle delle fasi `Fold`/`Unfold`.

Per migliorare le prestazioni e ridurre i costi, è importante provare a formulare trasversali in modo che la massima quantità di lavoro possibile venga eseguita nativamente all'interno del motore di query Neptune, anziché implementazioni graduali. TinkerPop

Esempio di una query che utilizza Neptune full-text-search

La query seguente utilizza la ricerca full-text Neptune:

```
g.withSideEffect("Neptune#fts.endpoint", "some_endpoint")
  .V()
  .tail(100)
  .has("Neptune#fts mark*")
  -------
  .has("name", "Neptune#fts mark*")
  .has("Person", "name", "Neptune#fts mark*")
```

La parte `.has("name", "Neptune#fts mark*")` limita la ricerca ai vertici con `name`, mentre `.has("Person", "name", "Neptune#fts mark*")` limita la ricerca ai vertici con `name` e all'etichetta `Person`. Ciò ha come risultato il seguente attraversamento nel report di `explain`:

```
Final Traversal
[NeptuneGraphQueryStep(Vertex) {
    JoinGroupNode {
        PatternNode[(?1, termid(1,URI), ?2, termid(0,URI)) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
    }, annotations={path=[Vertex(?1):GraphStep], maxVarId=4}
}, NeptuneTraverserConverterStep, NeptuneTailGlobalStep(10),
 NeptuneTinkerpopTraverserConverterStep, NeptuneSearchStep {
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=some_endpoint}
    }
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=some_endpoint}
    }
}]
```

Esempio di utilizzo di **`explain`** quando il motore DFE è abilitato

Di seguito è riportato un esempio di un report di `explain` quando il motore di query alternativo DFE è abilitato:

```
****************************************************
                Neptune Gremlin Explain
****************************************************

Query String
============

g.V().as("a").out().has("name", "josh").out().in().where(eq("a"))


Original Traversal
==================
[GraphStep(vertex,[])@[a], VertexStep(OUT,vertex), HasStep([name.eq(josh)]),
 VertexStep(OUT,vertex), VertexStep(IN,vertex), WherePredicateStep(eq(a))]

Converted Traversal
===================
Neptune steps:
[
    DFEStep(Vertex) {
      DFENode {
        DFEJoinGroupNode[ children={
          DFEPatternNode[(?1, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, ?2,
 <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>) . project DISTINCT[?1]
 {rangeCountEstimate=unknown}],
          DFEPatternNode[(?1, ?3, ?4, ?5) . project ALL[?1, ?4] graphFilters=(!
= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}]
        }, {rangeCountEstimate=unknown}
        ]
      } [Vertex(?1):GraphStep@[a], Vertex(?4):VertexStep]
    } ,
    NeptuneTraverserConverterDFEStep
]
+ not converted into Neptune steps: HasStep([name.eq(josh)]),
Neptune steps:
[
    NeptuneInterleavingStep {
      StepInfo[joinVars=[?7, ?1], frontierElement=Vertex(?7):HasStep,
 pathElements={a=(last,Vertex(?1):GraphStep@[a])}, listPathElement={}, indexTime=0ms],
      DFEStep(Vertex) {
        DFENode {
          DFEJoinGroupNode[ children={
```

```
            DFEPatternNode[(?7, ?8, ?9, ?10) . project ALL[?7, ?9]
 graphFilters=(!= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}],
            DFEPatternNode[(?12, ?11, ?9, ?13) . project ALL[?9, ?12]
 graphFilters=(!= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}]
          }, {rangeCountEstimate=unknown}
          ]
        } [Vertex(?9):VertexStep, Vertex(?12):VertexStep]
      }
    }
]
+ not converted into Neptune steps: WherePredicateStep(eq(a)),
Neptune steps:
[
    DFECleanupStep
]


Optimized Traversal
===================
Neptune steps:
[
    DFEStep(Vertex) {
      DFENode {
        DFEJoinGroupNode[ children={
          DFEPatternNode[(?1, ?3, ?4, ?5) . project ALL[?1, ?4] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=92233720368547758O7}]
        }, {rangeCountEstimate=unknown}
        ]
      } [Vertex(?1):GraphStep@[a], Vertex(?4):VertexStep]
    } ,
    NeptuneTraverserConverterDFEStep
]
+ not converted into Neptune steps: NeptuneHasStep([name.eq(josh)]),
Neptune steps:
[
    NeptuneMemoryTrackerStep,
    NeptuneInterleavingStep {
      StepInfo[joinVars=[?7, ?1], frontierElement=Vertex(?7):HasStep,
 pathElements={a=(last,Vertex(?1):GraphStep@[a])}, listPathElement={}, indexTime=0ms],
      DFEStep(Vertex) {
        DFENode {
          DFEJoinGroupNode[ children={
```

```
            DFEPatternNode[(?7, ?8, ?9, ?10) . project ALL[?7, ?9] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807}],
            DFEPatternNode[(?12, ?11, ?9, ?13) . project ALL[?9, ?12] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807}]
          }, {rangeCountEstimate=unknown}
          ]
        } [Vertex(?9):VertexStep, Vertex(?12):VertexStep]
      }
    }
]
+ not converted into Neptune steps: WherePredicateStep(eq(a)),
Neptune steps:
[
    DFECleanupStep
]


WARNING: >> [NeptuneHasStep([name.eq(josh)]), WherePredicateStep(eq(a))] << (or one of
 the children for each step) is not supported natively yet

Predicates
==========
# of predicates: 8
```

Per una descrizione delle sezioni specifiche del motore DFE nel report, vedi [Informazioni in `explain`](#).

API Gremlin **`profile`** in Neptune

L'API `profile` di Gremlin in Neptune esegue un attraversamento Gremlin specificato, raccoglie varie metriche relative all'esecuzione e produce un report di profile come output.

> **Note**
>
> Questa funzionalità è disponibile a partire dal [Rilascio 1.0.1.0.200463.0 (15/10/2019)](#).

Si differenzia dal passaggio TinkerPop .profile () per poter riportare informazioni specifiche per il motore Neptune.

Il report del profilo include le seguenti informazioni relative al piano di query:

- La pipeline dell'operatore fisico
- Le operazioni di indice per l'esecuzione e la serializzazione di query
- Le dimensioni del risultato

L'API `profile` utilizza una versione estesa della sintassi API HTTP per le query, con `/gremlin/profile` come endpoint anziché `/gremlin`.

Parametri specifici di **`profile`** di Gremlin in Neptune

- profile.results: `boolean`, valori consentiti: `TRUE` e `FALSE`, valore predefinito: `TRUE`.

  Se true, i risultati della query vengono raccolti e visualizzati come parte del report di `profile`. Se false, viene visualizzato solo il conteggio dei risultati.
- profile.chop: `int`, valore predefinito: 250.

  Se diverso da zero, la stringa dei risultati viene troncata a tale numero di caratteri. Ciò non impedisce l'acquisizione di tutti i risultati. Limita semplicemente le dimensioni della stringa nel report del profilo. Se impostato su zero, la stringa contiene tutti i risultati.
- profile.serializer: `string`, valore predefinito: `<null>`.

  Se diverso da null, i risultati raccolti vengono restituiti in un messaggio di risposta serializzato nel formato specificato da questo parametro. Il numero di operazioni di indice necessarie per produrre tale messaggio di risposta viene segnalato insieme alle dimensioni in byte da inviare al client.

  I valori consentiti sono `<null>` o uno qualsiasi dei valori enumerativi «Serializers» del TinkerPop driver o del tipo MIME validi.

```
"application/json"                       or  "MIME_JSON"
"application/vnd.gremlin-v1.0+json"      or  "GRAPHSON_V1D0"
"application/vnd.gremlin-v2.0+json"      or  "GRAPHSON_V2D0"
"application/vnd.gremlin-v3.0+json"      or  "GRAPHSON_V3D0"
"application/vnd.gremlin-v1.0+gryo"      or  "GRYO_V1D0"
"application/vnd.gremlin-v3.0+gryo"      or  "GRYO_V3D0"
"application/vnd.gremlin-v1.0+gryo-lite" or  "GRYO_LITE_V1D0"
"application/vnd.graphbinary-v1.0"       or  "GRAPHBINARY_V1D0"
```

- profile.indexOps: `boolean`, valori consentiti: `TRUE` e `FALSE`, valore predefinito: `FALSE`.

Se true, mostra un report dettagliato di tutte le operazioni sugli indici eseguite durante l'esecuzione e la serializzazione delle query. Avviso: questo report può essere dettagliato.

Output di esempio di **`profile`** di Gremlin in Neptune

Di seguito è riportata una query `profile` di esempio.

```
curl -X POST https://your-neptune-endpoint:port/gremlin/profile \
     -d '{"gremlin":"g.V().hasLabel(\"airport\")
                         .has(\"code\", \"AUS\")
                         .emit()
                         .repeat(in().simplePath())
                         .times(2)
                         .limit(100)",
          "profile.serializer":"application/vnd.gremlin-v3.0+gryo"}'
```

Questa query genera il seguente report `profile` quando eseguita sul grafo di esempio air-routes da post del blog, Let Me Graph That For You - Part 1 - Air Routes.

```
*******************************************************
                Neptune Gremlin Profile
*******************************************************

Query String
==================
g.V().hasLabel("airport").has("code",
 "AUS").emit().repeat(in().simplePath()).times(2).limit(100)

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([~label.eq(airport), code.eq(AUS)]),
 RepeatStep(emit(true),[VertexStep(IN,vertex), PathFilterStep(simple),
 RepeatEndStep],until(loops(2))), RangeGlobalStep(0,100)]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
```

```
            PatternNode[(?1, <code>, "AUS", ?) . project ?1 .],
 {estimatedCardinality=1, indexTime=84, hashJoin=true, joinTime=3, actualTotalOutput=1}
            PatternNode[(?1, <~label>, ?2=<airport>, <~>) . project ask .],
 {estimatedCardinality=3374, indexTime=29, hashJoin=true, joinTime=0,
 actualTotalOutput=61}
            RepeatNode {
                Repeat {
                    PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .
 SimplePathFilter(?1, ?3)) .], {hashJoin=true, estimatedCardinality=50148, indexTime=0,
 joinTime=3}
                }
                Emit {
                    Filter(true)
                }
                LoopsCondition {
                    LoopsFilter([?1, ?3],eq(2))
                }
            }, annotations={repeatMode=BFS, emitFirst=true, untilFirst=false, leftVar=?
1, rightVar=?3}
        }, finishers=[limit(100)], annotations={path=[Vertex(?1):GraphStep,
 Repeat[Vertex(?3):VertexStep]], joinStats=true, optimizationTime=495, maxVarId=7,
 executionTime=323}
    },
    NeptuneTraverserConverterStep
]

Physical Pipeline
=================
NeptuneGraphQueryStep
    |-- StartOp
    |-- JoinGroupOp
        |-- SpoolerOp(100)
        |-- DynamicJoinOp(PatternNode[(?1, <code>, "AUS", ?) . project ?1 .],
 {estimatedCardinality=1, indexTime=84, hashJoin=true})
        |-- SpoolerOp(100)
        |-- DynamicJoinOp(PatternNode[(?1, <~label>, ?2=<airport>, <~>) . project
 ask .], {estimatedCardinality=3374, indexTime=29, hashJoin=true})
        |-- RepeatOp
            |-- <upstream input> (Iteration 0) [visited=1, output=1 (until=0, emit=1),
 next=1]
            |-- BindingSetQueue (Iteration 1) [visited=61, output=61 (until=0,
 emit=61), next=61]
                |-- SpoolerOp(100)
```

```
                |-- DynamicJoinOp(PatternNode[(?3, ?5, ?1, ?6) . project ?
1,?3 . IsEdgeIdFilter(?6) . SimplePathFilter(?1, ?3)) .], {hashJoin=true,
 estimatedCardinality=50148, indexTime=0})
            |-- BindingSetQueue (Iteration 2) [visited=38, output=38 (until=38,
 emit=0), next=0]
                |-- SpoolerOp(100)
                |-- DynamicJoinOp(PatternNode[(?3, ?5, ?1, ?6) . project ?
1,?3 . IsEdgeIdFilter(?6) . SimplePathFilter(?1, ?3)) .], {hashJoin=true,
 estimatedCardinality=50148, indexTime=0})
        |-- LimitOp(100)

Runtime (ms)
============
Query Execution:  392.686
Serialization:   2636.380

Traversal Metrics
=================
Step                                                               Count  Traversers
    Time (ms)    % Dur
-------------------------------------------------------------------------------------------------
NeptuneGraphQueryStep(Vertex)                                        100         100
      314.162    82.78
NeptuneTraverserConverterStep                                        100         100
       65.333    17.22
                                              >TOTAL                   -           -
      379.495        -

Repeat Metrics
==============
Iteration  Visited   Output    Until     Emit     Next
------------------------------------------------------
        0        1        1        0        1        1
        1       61       61        0       61       61
        2       38       38       38        0        0
------------------------------------------------------
               100      100       38       62       62

Predicates
==========
# of predicates: 16

WARNING: reverse traversal with no edge label(s) - .in() / .both() may impact query
 performance
```

```
Results
=======
Count: 100
Output: [v[3], v[3600], v[3614], v[4], v[5], v[6], v[7], v[8], v[9], v[10], v[11],
 v[12], v[47], v[49], v[136], v[13], v[15], v[16], v[17], v[18], v[389], v[20], v[21],
 v[22], v[23], v[24], v[25], v[26], v[27], v[28], v[416], v[29], v[30], v[430], v[31],
 v[9...
Response serializer: GRYO_V3D0
Response size (bytes): 23566

Index Operations
================
Query execution:
    # of statement index ops: 3
    # of unique statement index ops: 3
    Duplication ratio: 1.0
    # of terms materialized: 0
Serialization:
    # of statement index ops: 200
    # of unique statement index ops: 140
    Duplication ratio: 1.43
    # of terms materialized: 393
```

Oltre ai piani di query restituiti da una chiamata a Neptune `explain`, i risultati di `profile` includono le statistiche di runtime relative all'esecuzione della query. Ogni operazione join è contrassegnata con il tempo richiesto per eseguire il join e il numero effettivo di soluzioni trasferite.

L'output `profile` include il tempo richiesto durante la fase di esecuzione della query principale, nonché la fase di serializzazione se l'opzione `profile.serializer` è stata specificata.

Nella parte inferiore dell'output `profile` è inclusa anche l'analisi dettagliata delle operazioni sugli indici eseguite durante ogni fase.

Notare che esecuzioni consecutive della stessa query possono mostrare risultati diversi in termini di tempo di esecuzione e operazioni sugli indici a causa del caching.

Per le query che utilizzano la fase `repeat()`, è disponibile un'analisi dettagliata della frontiera per ogni iterazione se la fase `repeat()` è stata trasferita come parte di una `NeptuneGraphQueryStep`.

Differenze nei report di **`profile`** quando è abilitato il motore DFE

Quando il motore di query alternativo Neptune DFE è abilitato, l'output di `profile` è leggermente diverso:

Attraversamento ottimizzato: questa sezione è simile a quella dell'output di `explain` ma contiene informazioni aggiuntive. Include il tipo di operatori DFE che sono stati considerati nella pianificazione e le stime dei costi associati nel caso peggiore e nel caso migliore.

Pipeline fisica: questa sezione raccoglie gli operatori utilizzati per eseguire la query. Gli elementi `DFESubQuery` astraggono il piano fisico utilizzato dal motore DFE per eseguire la parte del piano di cui è responsabile. Gli elementi `DFESubQuery` sono illustrati nella sezione seguente in cui sono elencate le statistiche del motore DFE.

QueryEngine Statistiche DFE: questa sezione viene visualizzata solo quando almeno una parte della query viene eseguita da DFE. Descrive varie statistiche di runtime specifiche di DFE e contiene una suddivisione dettagliata del tempo impiegato nelle varie parti dell'esecuzione della query, da parte di `DFESubQuery`.

Le sottoquery annidate in diversi elementi `DFESubQuery` sono semplificate in questa sezione e gli identificatori univoci sono contrassegnati da un'intestazione che inizia con `subQuery=`.

Metriche di attraversamento: questa sezione mostra le metriche di attraversamento a livello di passaggio e, quando il motore DFE esegue tutta o parte della query, visualizza le metriche per `DFEStep` e/o `NeptuneInterleavingStep`. Per informazioni, consulta Ottimizzazione delle query Gremlin con `explain` e `profile`.

> **Note**
>
> DFE è una funzionalità sperimentale rilasciata in modalità di laboratorio, quindi il formato esatto dell'output di `profile` è ancora soggetto a modifiche.

Output di **`profile`** di esempio quando il motore Neptune Dataflow (DFE) è abilitato

Quando il motore DFE viene utilizzato per eseguire le query Gremlin, l'output di API Gremlin `profile` è formattato come mostrato nell'esempio seguente.

Query:

```
curl https://localhost:8182/gremlin/profile \
  -d "{\"gremlin\": \"g.withSideEffect('Neptune#useDFE', true).V().has('code',
 'ATL').out()\"}"
```

```
*******************************************************
                    Neptune Gremlin Profile
    *******************************************************

    Query String
    ==================
    g.withSideEffect('Neptune#useDFE', true).V().has('code', 'ATL').out()

    Original Traversal
    ==================
    [GraphStep(vertex,[]), HasStep([code.eq(ATL)]), VertexStep(OUT,vertex)]

    Optimized Traversal
    ===================
    Neptune steps:
    [
        DFEStep(Vertex) {
          DFENode {
            DFEJoinGroupNode[null](
              children=[
                DFEPatternNode((?1, vp://code[419430926], ?4, defaultGraph[526]) .
 project DISTINCT[?1] objectFilters=(in(ATL[452987149]) . ), {rangeCountEstimate=1},
                    opInfo=(type=PipelineJoin,
 cost=(exp=(in=1.00,out=1.00,io=0.00,comp=0.00,mem=0.00),wc=(in=1.00,out=1.00,io=0.00,comp=0.00
                    disc=(type=PipelineScan,
 cost=(exp=(in=1.00,out=1.00,io=0.00,comp=0.00,mem=34.00),wc=(in=1.00,out=1.00,io=0.00,comp=0.0
                DFEPatternNode((?1, ?5, ?6, ?7) . project ALL[?1, ?6] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807})],
              opInfo=[
                OperatorInfoWithAlternative[
                  rec=(type=PipelineJoin,
 cost=(exp=(in=1.00,out=27.76,io=0.00,comp=0.00,mem=0.00),wc=(in=1.00,out=27.76,io=0.00,comp=0.
                    disc=(type=PipelineScan,
 cost=(exp=(in=1.00,out=27.76,io=Infinity,comp=0.00,mem=295147905179352830000.00),wc=(in=1.00,o
                    alt=(type=PipelineScan,
 cost=(exp=(in=1.00,out=27.76,io=Infinity,comp=0.00,mem=295147905179352830000.00),wc=(in=1.00,o
          } [Vertex(?1):GraphStep, Vertex(?6):VertexStep]
        } ,
        NeptuneTraverserConverterDFEStep,
```

```
        DFECleanupStep
    ]


    Physical Pipeline
    =================
    DFEStep
        |-- DFESubQuery1

    DFEQueryEngine Statistics
    =================
    DFESubQuery1

 ########################################################################################################
    # ID # Out #1 # Out #2 # Name                   # Arguments
                                                                                          # Mode #
 Units In # Units Out # Ratio  # Time (ms) #

 ########################################################################################################
    # 0  # 1      # -      # DFESolutionInjection  # solutions=[]
                                                                                        # -    # 0
      # 1         # 0.00   # 0.01      #
    #    #        #        #                       # outSchema=[]
                                                                                        #      #
      #           #        #           #

 ########################################################################################################
    # 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_1 # -    #
 1        # 1         # 1.00   # 0.02      #

 ########################################################################################################
    # 2  # 3      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_2 # -    #
 1        # 242       # 242.00 # 0.02      #

 ########################################################################################################
    # 3  # 4      # -      # DFEMergeChunks        # -
                                                                                        # -    # 242
      # 242       # 1.00   # 0.01      #

 ########################################################################################################
```

```
    # 4  # -      # -      # DFEDrain              # -
                                                                  # -    # 242
      # 0         # 0.00   # 0.01      #

 ##########################################################################################


    subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/
graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_1

 ##########################################################################################
    # ID # Out #1 # Out #2 # Name                 # Arguments
                          # Mode # Units In # Units Out # Ratio # Time (ms) #

 ##########################################################################################
    # 0  # 1      # -      # DFEPipelineScan      # pattern=Node(?1) with property
 'code' as ?4 and label 'ALL' # -    # 0        # 1         # 0.00  # 0.22      #
    #    #        #        #                     # inlineFilters=[(?4 IN ["ATL"])]
                          #      #          #           #       #           #
    #    #        #        #                     # patternEstimate=1
                          #      #          #           #       #           #

 ##########################################################################################
    # 1  # 2      # -      # DFEMergeChunks       # -
                          # -    # 1        # 1         # 1.00  # 0.02      #

 ##########################################################################################
    # 2  # 4      # -      # DFERelationalJoin    # joinVars=[]
                          # -    # 2        # 1         # 0.50  # 0.09      #

 ##########################################################################################
    # 3  # 2      # -      # DFESolutionInjection # solutions=[]
                          # -    # 0        # 1         # 0.00  # 0.01      #
    #    #        #        #                     # outSchema=[]
                          #      #          #           #       #           #

 ##########################################################################################
    # 4  # -      # -      # DFEDrain             # -
                          # -    # 1        # 0         # 0.00  # 0.01      #

 ##########################################################################################
```

```
    subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/
graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_2

 ##########################################################################################
    # ID # Out #1 # Out #2 # Name                 # Arguments
 # Mode # Units In # Units Out # Ratio  # Time (ms) #

 ##########################################################################################
    # 0  # 1      # -      # DFESolutionInjection # solutions=[]
 # -    # 0        # 1         # 0.00   # 0.01      #
    #    #        #        #                      # outSchema=[?1]
 #      #          #           #        #           #

 ##########################################################################################
    # 1  # 2      # 3      # DFETee               # -
 # -    # 1        # 2         # 2.00   # 0.01      #

 ##########################################################################################
    # 2  # 4      # -      # DFEDistinctColumn    # column=?1
 # -    # 1        # 1         # 1.00   # 0.21      #
    #    #        #        #                      # ordered=false
 #      #          #           #        #           #

 ##########################################################################################
    # 3  # 5      # -      # DFEHashIndexBuild    # vars=[?1]
 # -    # 1        # 1         # 1.00   # 0.03      #

 ##########################################################################################
    # 4  # 5      # -      # DFEPipelineJoin      # pattern=Edge((?1)-[?7:?5]->(?6))
 # -    # 1        # 242       # 242.00 # 0.51      #
    #    #        #        #                      # constraints=[]
 #      #          #           #        #           #
    #    #        #        #                      # patternEstimate=9223372036854775807
 #      #          #           #        #           #

 ##########################################################################################
    # 5  # 6      # 7      # DFESync              # -
 # -    # 243      # 243       # 1.00   # 0.02      #

 ##########################################################################################
    # 6  # 8      # -      # DFEForwardValue      # -
 # -    # 1        # 1         # 1.00   # 0.01      #

 ##########################################################################################
```

```
   # 7  # 8      # -      # DFEForwardValue      # -
# -    # 242      # 242       # 1.00   # 0.02      #

##################################################################################################
   # 8  # 9      # -      # DFEHashIndexJoin     # -
# -    # 243      # 242       # 1.00   # 0.31      #

##################################################################################################
   # 9  # -      # -      # DFEDrain             # -
# -    # 242      # 0         # 0.00   # 0.01      #

##################################################################################################


   Runtime (ms)
   ============
   Query Execution: 11.744

   Traversal Metrics
   =================
   Step                                                               Count
Traversers       Time (ms)    % Dur

-------------------------------------------------------------------------------------------------
   DFEStep(Vertex)                                                    242
242          10.849    95.48
   NeptuneTraverserConverterDFEStep                                   242
242           0.514     4.52
                                                >TOTAL                     -
-          11.363        -

   Predicates
   ==========
   # of predicates: 18

   Results
   =======
   Count: 242


   Index Operations
   ================
   Query execution:
       # of statement index ops: 0
```

```
        # of terms materialized: 0
```

> **Note**
>
> Poiché il motore DFE è una funzionalità sperimentale rilasciata in modalità di laboratorio, il formato esatto dell'output di `profile` è soggetto a modifiche.

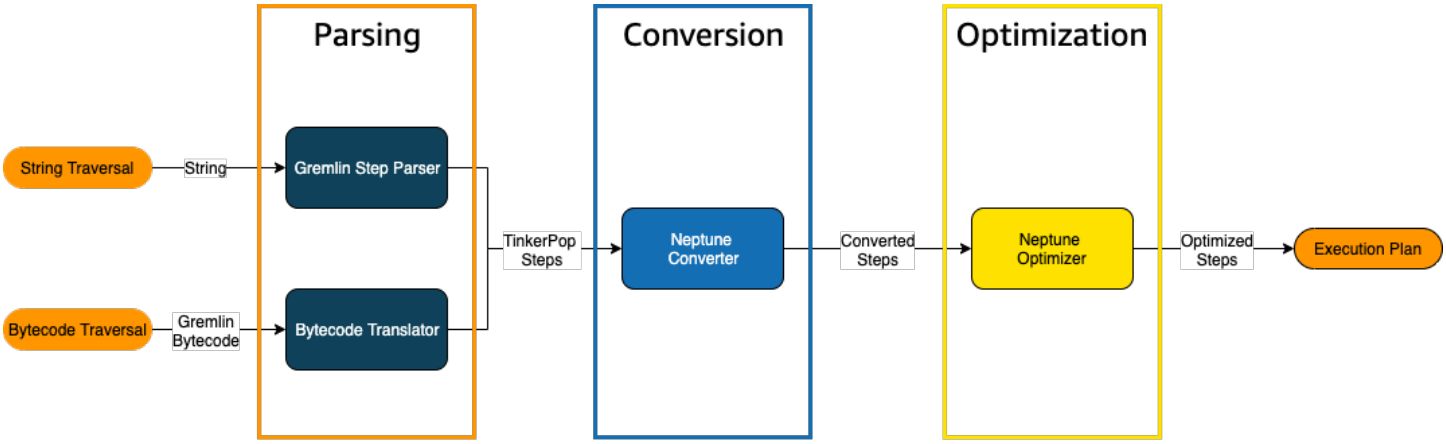
Ottimizzazione delle query Gremlin con **`explain`** e **`profile`**

Spesso è possibile ottimizzare le query Gremlin in Amazon Neptune per ottenere prestazioni migliori, utilizzando le informazioni disponibili nei report ottenuti dalle API explain e profile di Neptune. A tale scopo, è utile capire come Neptune elabora gli attraversamenti di Gremlin.

> **Important**
>
> Nella TinkerPop versione 3.4.11 è stata apportata una modifica che migliora la correttezza del modo in cui le query vengono elaborate, ma per il momento a volte può influire seriamente sulle prestazioni delle query.
> Una query di questo tipo, ad esempio, può essere eseguita molto più lentamente:
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
>   out()
> ```
>
> I vertici dopo il passaggio limite vengono ora recuperati in modo non ottimale a causa della modifica 3.4.11. TinkerPop Per evitare il problema, puoi modificare la query aggiungendo il passaggio barrier() in qualsiasi punto dopo `order().by()`. Per esempio:
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
>   barrier().
>   out()
> ```
>
> TinkerPop 3.4.11 è stato abilitato nella versione 1.0.5.0 del motore Neptune.

Informazioni sull'elaborazione degli attraversamenti di Gremlin in Neptune

Quando un attraversamento Gremlin viene inviato a Neptune, ci sono tre processi principali che trasformano l'attraversamento in un piano di esecuzione sottostante che il motore deve eseguire. Questi sono l'analisi, la conversione e l'ottimizzazione:



Processo di analisi di un attraversamento

Il primo passaggio nell'elaborazione di un attraversamento consiste nell'analizzarlo in un linguaggio comune. In Neptune, quel linguaggio comune è l'insieme TinkerPop di passaggi che fanno parte dell'API. TinkerPop Ciascuno di questi passaggi rappresenta un'unità di calcolo all'interno dell'attraversamento.

È possibile inviare un attraversamento Gremlin a Neptune come stringa o come bytecode. L'endpoint REST e il metodo `submit()` del driver client Java inviano gli attraversamenti come stringhe, come in questo esempio:

```
client.submit("g.V()")
```

Le applicazioni e i driver di linguaggio che utilizzano le varianti del linguaggio Gremlin (GLV) inviano attraversamenti in bytecode.

Processo di conversione di un attraversamento

Il secondo passaggio nell'elaborazione di un attraversamento consiste nel convertire TinkerPop i suoi passaggi in un insieme di passaggi di Nettuno convertiti e non convertiti. La maggior parte dei passaggi del linguaggio di interrogazione Apache TinkerPop Gremlin viene convertita in passaggi specifici di Neptune ottimizzati per l'esecuzione sul motore Neptune sottostante. Quando si incontra un TinkerPop passaggio senza un equivalente di Neptune in un attraversamento, quel passaggio e tutti i passaggi successivi dell'attraversamento vengono elaborati dal motore di query. TinkerPop

Per ulteriori informazioni su quali passaggi possono essere convertiti e in quali circostanze, consulta Supporto dei passaggi Gremlin.

Processo di ottimizzazione di un attraversamento

Il passaggio finale dell'elaborazione di un attraversamento consiste nell'eseguire la serie di passaggi convertiti e non convertiti tramite l'ottimizzatore, per cercare di determinare il miglior piano di esecuzione. Il risultato di questa ottimizzazione è il piano di esecuzione elaborato dal motore Neptune.

Utilizzo dell'API **`explain`** di Gremlin in Neptune per ottimizzare le query

L'API explain di Neptune non è la stessa del passaggio `explain()` di Gremlin. Restituisce il piano di esecuzione finale che il motore Neptune elaborerà durante l'esecuzione della query. Poiché non esegue alcuna elaborazione, restituisce lo stesso piano indipendentemente dai parametri utilizzati e l'output non contiene statistiche sull'esecuzione effettiva.

Considerare il semplice attraversamento seguente che trova tutti i vertici dell'aeroporto di Anchorage:

```
g.V().has('code','ANC')
```

Esistono due modi per eseguire questo attraversamento tramite l'API Neptune `explain`. Il primo modo è effettuare una chiamata REST all'endpoint explain, come riportato di seguito:

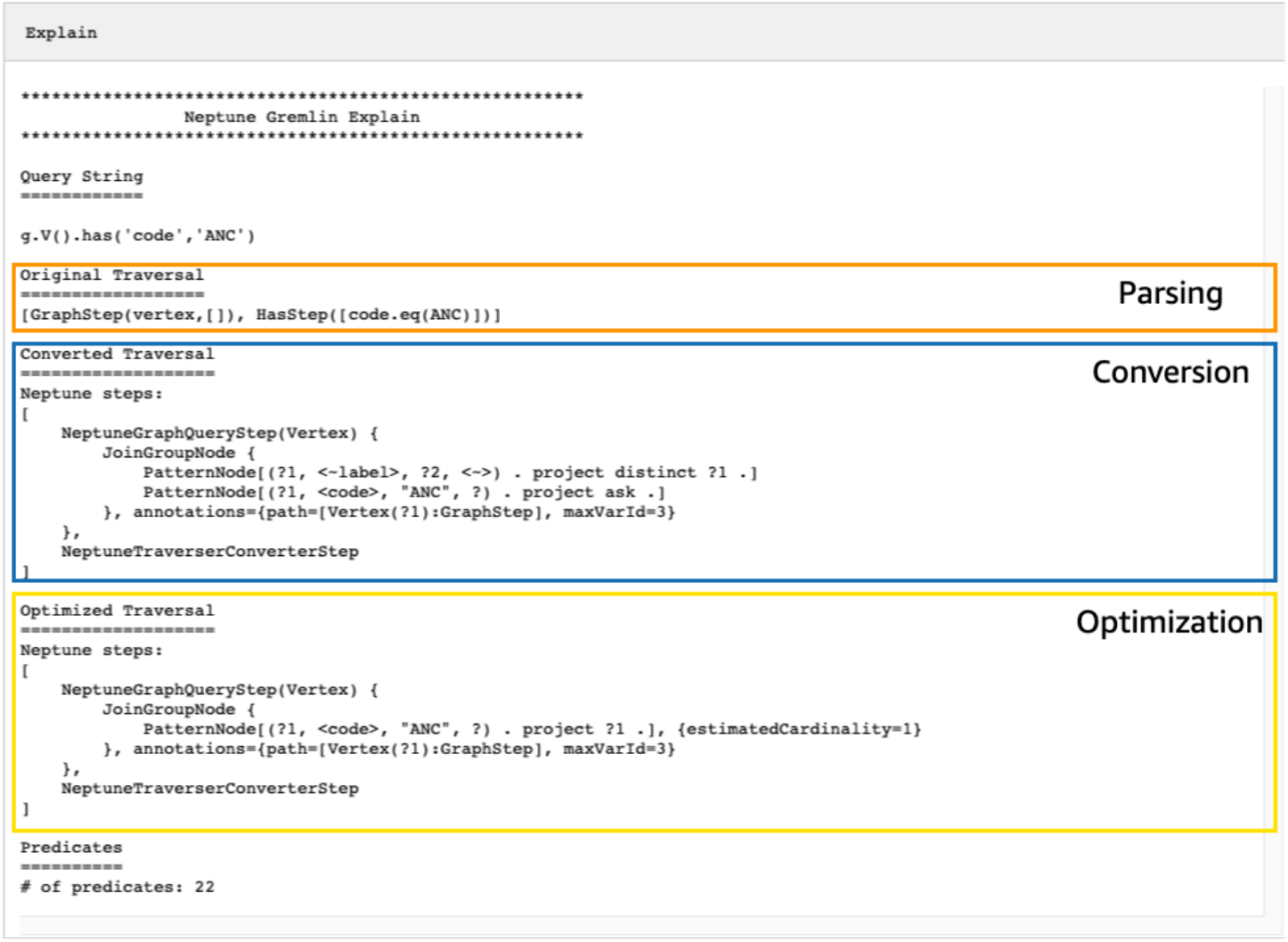
```
curl -X POST https://your-neptune-endpoint:port/gremlin/explain -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

Il secondo modo consiste nell'utilizzare il comando magic di cella %%gremlin di Neptune Workbench con il parametro `explain`. In questo modo si passa l'attraversamento contenuto nel corpo della cella all'API Neptune `explain` e quindi si visualizza l'output risultante quando si esegue la cella:

```
%%gremlin explain

g.V().has('code','ANC')
```

L'output dell'API `explain` risultante descrive il piano di esecuzione di Neptune per l'attraversamento. Come si può vedere nell'immagine seguente, il piano include ognuno dei 3 passaggi della pipeline di elaborazione:



Ottimizzazione di un attraversamento osservando i passaggi non convertiti

Una delle prime cose da verificare nell'output dell'API Neptune `explain` riguarda la presenza di passaggi Gremlin non convertiti in passaggi nativi Neptune. In un piano di query, quando viene rilevato un passaggio che non può essere convertito in un passaggio nativo Neptune, questo passaggio e tutti i passaggi successivi del piano vengono elaborati dal server Gremlin.

Nell'esempio riportato sopra, tutti i passaggi dell'attraversamento sono stati convertiti. Esaminiamo l'output dell'API `explain` per questo attraversamento:

```
g.V().has('code','ANC').out().choose(hasLabel('airport'), values('code'), constant('Not
 an airport'))
```

Come si può vedere nell'immagine qui sotto, Neptune non è riuscito a convertire il passaggio `choose()`:

```
Explain
```

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============

g.V().has('code','ANC').out().choose(hasLabel('airport'), values('code'), constant('Not an airport'))

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(OUT,vertex), ChooseStep([HasStep([~label.eq(airport)]), HasNex

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [ChooseStep([HasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [PropertiesSte

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .], {estimatedCardinality=1}
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .], {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [ChooseStep([NeptuneHasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [Proper

WARNING: >> ChooseStep([NeptuneHasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [PropertiesStep([code],value), E

Predicates
==========
# of predicates: 26
```

Ci sono diverse cose che si possono fare per ottimizzare le prestazioni dell'attraversamento. La prima sarebbe quella di riscriverlo in modo tale da eliminare il passaggio che non può essere convertito. Un'altra potrebbe essere quella di spostare il passaggio alla fine dell'attraversamento in modo che tutti gli altri passaggi possano essere convertiti in passaggi nativi.

Non è sempre necessario ottimizzare un piano di query con passaggi non convertiti. Se i passaggi che non possono essere convertiti si trovano alla fine dell'attraversamento e sono correlati al modo in cui viene formattato l'output anziché a come viene attraversato il grafo, possono avere un effetto minimo sulle prestazioni.

Un'altra cosa da considerare quando si esamina l'output dell'API Neptune `explain` sono i passaggi che non utilizzano indici. Il seguente attraversamento trova tutti gli aeroporti con voli che atterrano ad Anchorage:

```
g.V().has('code','ANC').in().values('code')
```

L'output dell'API explain per questo attraversamento è:

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============

g.V().has('code','ANC').in().values('code')

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(IN,vertex),
 PropertiesStep([code],value)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
            PatternNode[(?3, ?7, ?8, <~>) . project ?3,?8 . ContainsFilter(?7 in
 (<code>)) .]
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
```

```
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .],
 {estimatedCardinality=1}
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .],
 {estimatedCardinality=INFINITY}
            PatternNode[(?3, ?7=<code>, ?8, <~>) . project ?3,?8 .],
 {estimatedCardinality=7564}
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 26

WARNING: reverse traversal with no edge label(s) - .in() / .both() may impact query
 performance
```

Il messaggio `WARNING` nella parte inferiore dell'output è dovuto al fatto che il passaggio `in()` dell'attraversamento non può essere gestito utilizzando uno dei 3 indici gestiti da Neptune (vedi In che modo le dichiarazioni vengono indicizzate in Neptune e Dichiarazioni Gremlin in Neptune). Poiché il passaggio `in()` non contiene alcun filtro per gli archi, non può essere risolto utilizzando l'indice `SPOG`, `POGS` o `GPSO`. Neptune deve invece eseguire una scansione di unione per trovare i vertici richiesti, il che è molto meno efficiente.

Esistono due modi per ottimizzare l'attraversamento in questa situazione. Il primo consiste nell'aggiungere uno o più criteri di filtro al passaggio `in()` in modo da poter utilizzare una ricerca indicizzata per risolvere la query. Per l'esempio precedente, si avrà:

```
g.V().has('code','ANC').in('route').values('code')
```

L'output dell'API Neptune `explain` per l'attraversamento rivisto non contiene più il messaggio `WARNING`:

```
*******************************************************
                Neptune Gremlin Explain
```

```
****************************************************

Query String
============

g.V().has('code','ANC').in('route').values('code')

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(IN,[route],vertex),
 PropertiesStep([code],value)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .
 ContainsFilter(?5 in (<route>)) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
            PatternNode[(?3, ?7, ?8, <~>) . project ?3,?8 . ContainsFilter(?7 in
 (<code>)) .]
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .],
 {estimatedCardinality=1}
            PatternNode[(?3, ?5=<route>, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?
6) .], {estimatedCardinality=32042}
            PatternNode[(?3, ?7=<code>, ?8, <~>) . project ?3,?8 .],
 {estimatedCardinality=7564}
```

```
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 26
```

Un'altra opzione se si eseguono molti attraversamenti di questo tipo è quella di eseguirli in un cluster database Neptune con l'indice OSGP opzionale abilitato (vedi Abilitazione di un indice OSGP). L'abilitazione di un indice OSGP presenta degli svantaggi:

- Deve essere abilitato in un cluster database prima che vengano caricati i dati.
- La velocità di inserimento di vertici e archi può rallentare fino al 23%.
- L'utilizzo dello spazio di archiviazione aumenterà di circa il 20%.
- Le query di lettura che distribuiscono le richieste su tutti gli indici possono avere latenze maggiori.

Avere un indice OSGP è molto utile per un insieme limitato di modelli di query, ma a meno che non li si esegua frequentemente, in genere è preferibile cercare di garantire che gli attraversamenti scritti possano essere risolti utilizzando i tre indici primari.

Utilizzo di un numero elevato di predicati

Neptune considera ogni etichetta di arco e ogni nome distinto di proprietà di vertice o arco nel grafo come predicato ed è progettato per impostazione predefinita per funzionare con un numero relativamente basso di predicati distinti. Quando i dati del grafo contengono più di qualche migliaio di predicati, le prestazioni possono peggiorare.

L'output di Neptune `explain` avviserà se questo è il caso:

```
Predicates
==========
# of predicates: 9549
WARNING: high predicate count (# of distinct property names and edge labels)
```

Se non è conveniente rielaborare il modello di dati per ridurre il numero di etichette e proprietà, e quindi il numero di predicati, il modo migliore per ottimizzare gli attraversamenti è eseguirli in un cluster database con l'indice OSGP abilitato, come descritto in precedenza.

Utilizzo dell'API **`profile`** di Gremlin in Neptune per ottimizzare gli attraversamenti

L'API Neptune `profile` è molto diversa dal passaggio `profile()` di Gremlin. Come l'API `explain`, il relativo output include il piano di query utilizzato dal motore Neptune durante l'esecuzione dell'attraversamento. Inoltre, l'output di `profile` include le statistiche di esecuzione effettive per l'attraversamento, in base a come sono impostati i parametri.

Considerare anche in questo caso il semplice attraversamento che trova tutti i vertici dell'aeroporto di Anchorage:

```
g.V().has('code','ANC')
```

Come per l'API `explain`, è possibile richiamare l'API `profile` tramite una chiamata REST:

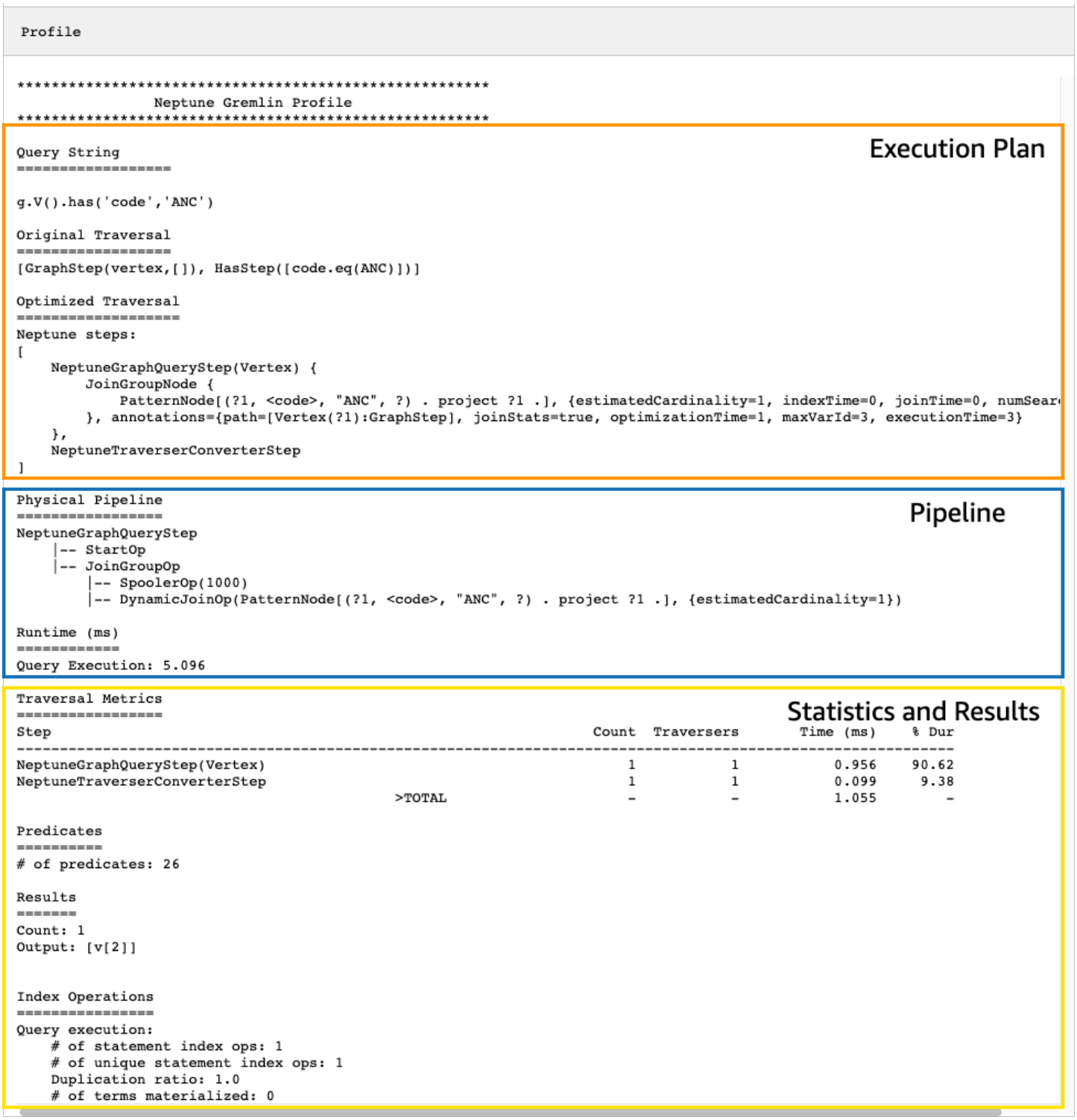
```
curl -X POST https://your-neptune-endpoint:port/gremlin/profile -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

È anche possibile utilizzare il comando magic di cella %%gremlin di Neptune Workbench con il parametro `profile`. In questo modo si passa l'attraversamento contenuto nel corpo della cella all'API Neptune `profile` e quindi si visualizza l'output risultante quando si esegue la cella:

```
%%gremlin profile

g.V().has('code','ANC')
```

L'output dell'API `profile` risultante contiene sia il piano di esecuzione di Neptune per l'attraversamento che le statistiche sull'esecuzione del piano, come si può vedere in questa immagine:



Nell'output di `profile`, la sezione del piano di esecuzione contiene solo il piano di esecuzione finale per l'attraversamento, non i passaggi intermedi. La sezione della pipeline contiene le operazioni della pipeline fisica che sono state eseguite e il tempo effettivo (in millisecondi) impiegato dall'esecuzione

dell'attraversamento. La metrica di runtime è estremamente utile per confrontare i tempi impiegati da due diverse versioni di un attraversamento durante l'ottimizzazione.

> **Note**
>
> Il runtime iniziale di un attraversamento è generalmente più lungo rispetto ai runtime successivi, poiché il primo fa sì che i dati pertinenti vengano memorizzati nella cache.

La terza sezione dell'output di `profile` contiene le statistiche di esecuzione e i risultati dell'attraversamento. Per capire come queste informazioni possano essere utili per ottimizzare un attraversamento, considerare il seguente attraversamento, che trova tutti gli aeroporti il cui nome inizia con "Anchora" e tutti gli aeroporti raggiungibili in due fermate da tali aeroporti, indicando i codici aeroportuali, le rotte di volo e le distanze:

```
%%gremlin profile

g.withSideEffect("Neptune#fts.endpoint", "{your-OpenSearch-endpoint-URL").
    V().has("city", "Neptune#fts Anchora~").
    repeat(outE('route').inV().simplePath()).times(2).
    project('Destination', 'Route').
        by('code').
        by(path().by('code').by('dist'))
```

Metriche di attraversamento nell'output dell'API Neptune **`profile`**

Il primo set di metriche disponibile in tutti gli output di `profile` è costituito dalle metriche di attraversamento. Sono simili alle metriche del passaggio `profile()` di Gremlin, con alcune differenze:

```
Traversal Metrics
=================
Step                                                               Count  Traversers
    Time (ms)    % Dur
-------------------------------------------------------------------------------------------------
NeptuneGraphQueryStep(Vertex)                                       3856        3856
       91.701     9.09
NeptuneTraverserConverterStep                                       3856        3856
       38.787     3.84
ProjectStep([Destination, Route],[value(code), ...                  3856        3856
      878.786    87.07
```

```
  PathStep([value(code), value(dist)])                                3856        3856
     601.359
                                                >TOTAL                    -           -
    1009.274        -
```

La prima colonna della tabella delle metriche di attraversamento elenca i passaggi eseguiti dall'attraversamento. I primi due passaggi sono generalmente i passaggi specifici di Neptune, `NeptuneGraphQueryStep` e `NeptuneTraverserConverterStep`.

`NeptuneGraphQueryStep` rappresenta il tempo di esecuzione per l'intera porzione dell'attraversamento che potrebbe essere convertito ed eseguito in modo nativo dal motore Neptune.

`NeptuneTraverserConverterStep`rappresenta il processo di conversione dell'output di tali passaggi convertiti in TinkerPop traverser che consentono di elaborare i passaggi che non possono essere convertiti, se presenti, o di restituire i risultati in un formato compatibile. TinkerPop

Nell'esempio precedente, abbiamo diversi passaggi non convertiti, quindi vediamo che ognuno di questi TinkerPop passaggi (`ProjectStep`,`PathStep`) appare quindi come una riga nella tabella.

La seconda colonna della tabella riporta il numero di traverser rappresentati che hanno superato il passaggio, mentre la terza colonna riporta il numero di traverser che hanno superato quel passaggio, come spiegato nella documentazione della fase del profilo. `CountTraversers` TinkerPop

Nell'esempio ci sono 3.856 vertici e 3.856 traverser restituiti da `NeptuneGraphQueryStep`, e questi numeri rimangono invariati per tutta l'elaborazione rimanente perché `ProjectStep` e `PathStep` stanno formattando i risultati, non li filtrano.

> **Note**
>
> Al contrario TinkerPop, il motore Neptune non ottimizza le prestazioni ingombrando i suoi passaggi. `NeptuneGraphQueryStep` `NeptuneTraverserConverterStep` Il bulking è l' TinkerPopoperazione che combina i trasversali sullo stesso vertice per ridurre il sovraccarico operativo, e questo è ciò che causa la differenza tra i numeri e. `Count Traversers` Poiché il bulking si verifica solo nelle fasi a cui Neptune è delegato e non nelle fasi gestite TinkerPop da Neptune in modo nativo, le colonne and raramente differiscono. `Count Traverser`

La colonna Time riporta il numero di millisecondi impiegato dal passaggio e la colonna `% Dur` riporta la percentuale del tempo di elaborazione totale impiegato dal passaggio. Queste sono le metriche

che indicano dove concentrare gli sforzi di ottimizzazione, mostrando i passaggi che hanno richiesto più tempo.

Metriche delle operazioni di indicizzazione nell'output dell'API Neptune **`profile`**

Un altro set di metriche nell'output dell'API Neptune profile è costituito dalle operazioni di indicizzazione:

```
Index Operations
================
Query execution:
    # of statement index ops: 23191
    # of unique statement index ops: 5960
    Duplication ratio: 3.89
    # of terms materialized: 0
```

Queste metriche riportano:

- Numero totale di ricerche nell'indice.
- Numero di ricerche univoche eseguite nell'indice.
- Rapporto tra le ricerche totali nell'indice e quelle univoche. Un rapporto più basso indica una minore ridondanza.
- Numero di termini materializzati dal dizionario dei termini.

Metriche di ripetizione nell'output dell'API Neptune **`profile`**

Se l'attraversamento utilizza un passaggio `repeat()` come nell'esempio precedente, nell'output di `profile` viene visualizzata una sezione contenente le metriche di ripetizione:

```
Repeat Metrics
==============
Iteration  Visited   Output    Until     Emit     Next
-------------------------------------------------------
        0        2        0        0        0        2
        1       53        0        0        0       53
        2     3856     3856     3856        0        0
-------------------------------------------------------
              3911     3856     3856        0       55
```

Queste metriche riportano:

- Conteggio dei cicli per una riga (colonna `Iteration`).
- Numero di elementi visitati dal ciclo (colonna `Visited`).
- Numero di elementi restituiti dal ciclo (colonna `Output`).
- Ultimo elemento restituito dal ciclo (colonna `Until`).
- Numero di elementi emessi dal ciclo (colonna `Emit`).
- Numero di elementi passati dal ciclo al ciclo successivo (colonna `Next`).

Queste metriche di ripetizione sono molto utili per comprendere il fattore di ramificazione dell'attraversamento e per avere un'idea di quanto lavoro venga svolto dal database. È possibile utilizzare questi numeri per diagnosticare problemi di prestazioni, soprattutto quando lo stesso attraversamento si comporta in modo notevolmente diverso con parametri diversi.

Metriche di ricerca full-text nell'output dell'API Neptune **`profile`**

Quando un attraversamento utilizza una ricerca full-text, come nell'esempio precedente, nell'output di `profile` viene visualizzata una sezione contenente le metriche di ricerca full-text (FTS):

```
FTS Metrics
==============
SearchNode[(idVar=?1, query=Anchora~, field=city) . project ?1 .],
    {endpoint=your-OpenSearch-endpoint-URL, incomingSolutionsThreshold=1000,
 estimatedCardinality=INFINITY,
    remoteCallTimeSummary=[total=65, avg=32.500000, max=37, min=28],
    remoteCallTime=65, remoteCalls=2, joinTime=0, indexTime=0, remoteResults=2}

    2 result(s) produced from SearchNode above
```

Questo mostra la query inviata al cluster ElasticSearch (ES) e riporta diverse metriche sull'interazione con le quali è possibile individuare i problemi di prestazioni relativi alla ricerca ElasticSearch di testo completo:

- Informazioni di riepilogo sulle chiamate all'indice: ElasticSearch
  - Numero totale di millisecondi richiesti da tutti gli oggetti remoteCalls per soddisfare la query (`total`).
  - Numero medio di millisecondi trascorsi in un oggetto remoteCall (`avg`).
  - Numero minimo di millisecondi trascorsi in un oggetto remoteCall (`min`).
  - Numero massimo di millisecondi trascorsi in un oggetto remoteCall (`max`).

- Tempo totale impiegato da RemoteCalls to ElasticSearch ()`remoteCallTime`.
- Il numero di chiamate RemoteCall effettuate a (). ElasticSearch `remoteCalls`
- Il numero di millisecondi spesi per unire i risultati (). ElasticSearch `joinTime`
- Numero di millisecondi trascorsi nelle ricerche nell'indice (`indexTime`).
- Il numero totale di risultati restituiti da (). ElasticSearch `remoteResults`

Supporto nativo dei passaggi Gremlin in Amazon Neptune

Il motore Amazon Neptune attualmente non dispone del supporto nativo completo di tutti i passaggi Gremlin, come spiegato in Ottimizzazione di query Gremlin. Il supporto attuale si divide in quattro categorie:

- Passaggi Gremlin che possono sempre essere convertiti in operazioni native del motore Neptune
- Passaggi Gremlin che possono essere convertiti in operazioni native del motore Neptune in alcuni casi
- Passaggi Gremlin che non vengono mai convertiti in operazioni native del motore Neptune
- Passaggi Gremlin che non sono affatto supportati in Neptune

Passaggi Gremlin che possono sempre essere convertiti in operazioni native del motore Neptune

Molti passaggi Gremlin possono essere convertiti in operazioni native del motore Neptune purché soddisfino le seguenti condizioni:

- Non devono essere preceduti nella query da un passaggio che non possa essere convertito.
- Il relativo passaggio padre, se presente, deve poter essere convertito.
- Tutti i relativi attraversamenti figlio, se presenti, devono poter essere convertiti.

I seguenti passaggi Gremlin vengono sempre convertiti in operazioni native del motore Neptune se soddisfino queste condizioni:

- and( )
- as( )
- count( )
- E( )

- emit( )
- explain( )
- group( )
- groupCount( )
- has( )
- identity( )
- is( )
- key( )
- label( )
- limit( )
- local( )
- loops( )
- not( )
- or( )
- profile( )
- properties( )
- subgraph( )
- until( )
- V( )
- value
- valueMap( )
- values( )

Passaggi Gremlin che possono essere convertiti in operazioni native del motore Neptune in alcuni casi

Alcuni passaggi Gremlin possono essere convertiti in operazioni native del motore Neptune in alcune situazioni ma non in altre:

- addE( ): il passaggio `addE()` può generalmente essere convertito in un'operazione nativa del motore Neptune, a meno che non sia immediatamente seguito da un passaggio `property()` contenente un attraversamento come chiave.

- addV( ): il passaggio `addV()` può generalmente essere convertito in un'operazione nativa del motore Neptune, a meno che non sia immediatamente seguito da un passaggio `property()` contenente un attraversamento come chiave o a meno che non vengano assegnate più etichette.
- aggregate( ): il passaggio `aggregate()` può generalmente essere convertito in un'operazione nativa del motore Neptune, a meno che il passaggio non venga utilizzato in un attraversamento figlio o sottoattraversamento oppure a meno che il valore archiviato non sia qualcosa di diverso da un valore di vertice, arco, ID, etichetta o proprietà.

  Nell'esempio seguente, `aggregate()` non viene convertito perché viene utilizzato in un attraversamento figlio:

  ```
  g.V().has('code','ANC').as('a')
       .project('flights').by(select('a')
       .outE().aggregate('x'))
  ```

  In questo esempio, aggregate() non viene convertito perché ciò che viene archiviato è il valore `min()` di un valore:

  ```
  g.V().has('code','ANC').outE().aggregate('x').by(values('dist').min())
  ```

- barrier( ): il passaggio `barrier()` può generalmente essere convertito in un'operazione nativa del motore Neptune, a meno che il passaggio successivo non venga convertito.
- cap( ): l'unico caso in cui il passaggio `cap()` viene convertito è quando viene combinato con il passaggio `unfold()` per restituire una versione espansa di un aggregato di valori di vertice, arco, ID o proprietà. In questo esempio, `cap()` verrà convertito perché è seguito da `.unfold()`:

  ```
  g.V().has('airport','country','IE').aggregate('airport').limit(2)
       .cap('airport').unfold()
  ```

  Tuttavia, se si rimuove `.unfold()`, `cap()` non verrà convertito:

  ```
  g.V().has('airport','country','IE').aggregate('airport').limit(2)
       .cap('airport')
  ```

- coalesce () — L'unico caso in cui il `coalesce()` passaggio viene convertito è quando segue lo schema Upsert consigliato nella pagina delle TinkerPop ricette. Altri modelli coalesce() non sono consentiti. La conversione è limitata al caso in cui tutti gli attraversamenti figlio possano essere

convertiti, producano tutti lo stesso tipo di output (vertice, arco, ID, valore, chiave o etichetta), attraversino tutti un nuovo elemento e non contengano il passaggio `repeat()`.

- constant( ): il passaggio constant() viene attualmente convertito solo se viene utilizzato all'interno di una parte `sack().by()` di un attraversamento per assegnare un valore costante, come questo:

```
g.V().has('code','ANC').sack(assign).by(constant(10)).out().limit(2)
```

- cyclicPath( ): il passaggio `cyclicPath()` può generalmente essere convertito in un'operazione nativa del motore Neptune, a meno che il passaggio non venga utilizzato con i modulatori `by()`, `from()` o `to()`. Nelle seguenti query, ad esempio, `cyclicPath()` non viene convertito:

```
g.V().has('code','ANC').as('a').out().out().cyclicPath().by('code')
g.V().has('code','ANC').as('a').out().out().cyclicPath().from('a')
g.V().has('code','ANC').as('a').out().out().cyclicPath().to('a')
```

- drop( ): il passaggio `drop()` può generalmente essere convertito in un'operazione nativa del motore Neptune, a meno che il passaggio non venga utilizzato all'interno di un passaggio `sideEffect()` o `optional()`.
- fold () — Ci sono solo due situazioni in cui il passaggio fold () può essere convertito, vale a dire quando viene utilizzato nel pattern Upsert consigliato nella pagina delle TinkerPop ricette e quando viene utilizzato in un contesto come questo: `group().by()`

```
g.V().has('code','ANC').out().group().by().by(values('code', 'city').fold())
```

- id( ): il passaggio `id()` viene convertito a meno che non venga utilizzato su una proprietà, come in questo caso:

```
g.V().has('code','ANC').properties('code').id()
```

- order( ): il passaggio `order()` può generalmente essere convertito in un'operazione nativa del motore Neptune, a meno che non si verifichi una delle seguenti condizioni:
  - Il passaggio `order()` si trova all'interno di un attraversamento figlio annidato, come questo:

```
g.V().has('code','ANC').where(V().out().order().by(id))
```

  - Viene utilizzato l'ordinamento locale, come ad esempio con `order(local)`.
  - Nella modulazione `by()` viene utilizzato un comparatore personalizzato per l'ordinamento. Un esempio è l'uso di `sack()` in questo modo:

```
g.withSack(0).
  V().has('code','ANC').
      repeat(outE().sack(sum).by('dist').inV()).times(2).limit(10).
      order().by(sack())
```

  - Esistono più ordinamenti sullo stesso elemento.

- project( ): il passaggio `project()` può generalmente essere convertito in un'operazione nativa del motore Neptune, a meno che il numero di istruzioni `by()` che segue `project()` non corrisponda al numero di etichette specificate, come in questo caso:

```
g.V().has('code','ANC').project('x', 'y').by(id)
```

- range( ): il passaggio `range()` viene convertito solo quando il limite inferiore dell'intervallo in questione è zero (ad esempio, `range(0,3)`).
- repeat( ): il passaggio `repeat()` può generalmente essere convertito in un'operazione nativa del motore Neptune, a meno che non sia annidato all'interno di un altro passaggio `repeat()`, come in questo caso:

```
g.V().has('code','ANC').repeat(out().repeat(out()).times(2)).times(2)
```

- sack( ): il passaggio `sack()` può generalmente essere convertito in un'operazione nativa del motore Neptune, tranne nei casi seguenti:
  - Se viene utilizzato un operatore sack non numerico.
  - Se viene utilizzato un operatore sack numerico diverso da `+`, `-`, `mult`, `div`, `min` e `max`.
  - Se `sack()` viene utilizzato all'interno di un passaggio `where()` per filtrare in base a un valore sack, come in questo caso:

```
g.V().has('code','ANC').sack(assign).by(values('code')).where(sack().is('ANC'))
```

- sum( ): il passaggio `sum()` può generalmente essere convertito in un'operazione nativa del motore Neptune, ma non quando viene utilizzato per calcolare una somma globale, come in questo caso:

```
g.V().has('code','ANC').outE('routes').values('dist').sum()
```

- union( ): il passaggio `union()` può essere convertito in un'operazione nativa del motore Neptune purché sia l'ultimo passaggio della query a parte il passaggio terminale.

- unfold () — Il `unfold()` passaggio può essere convertito in un funzionamento del motore Neptune nativo solo quando viene utilizzato nel pattern Upsert consigliato nella pagina TinkerPopdelle ricette e quando viene utilizzato insieme a questo: `cap()`

```
g.V().has('airport','country','IE').aggregate('airport').limit(2)
     .cap('airport').unfold()
```

- where( ): il passaggio `where()` può generalmente essere convertito in un'operazione nativa del motore Neptune, tranne nei casi seguenti:
  - Quando vengono utilizzate le modulazioni by(), come in questo caso:

```
g.V().hasLabel('airport').as('a')
     .where(gt('a')).by('runways')
```

  - Quando vengono utilizzati operatori di confronto diversi da `eq`, `neq`, `within` e `without`.
  - Quando vengono utilizzate aggregazioni fornite dall'utente.

Passaggi Gremlin che non vengono mai convertiti in operazioni native del motore Neptune

I seguenti passaggi Gremlin sono supportati in Neptune ma non vengono mai convertiti in operazioni native del motore Neptune. Vengono invece eseguiti dal server Gremlin.

- choose( )
- coin( )
- inject( )
- match( )
- math( )
- max( )
- mean( )
- min( )
- option( )
- optional( )
- path( )
- propertyMap( )
- sample( )

- skip( )
- tail( )
- timeLimit( )
- tree( )

Passaggi Gremlin che non sono affatto supportati in Neptune

I seguenti passaggi Gremlin non sono affatto supportati in Neptune. Nella maggior parte dei casi ciò è dovuto al fatto che richiedono un oggetto `GraphComputer`, che Neptune attualmente non supporta.

- connectedComponent( )
- io( )
- shortestPath( )
- withComputer( )
- pageRank( )
- peerPressure( )
- program( )

Il passaggio `io()` è in realtà parzialmente supportato, in quanto può essere utilizzato per eseguire un passaggio `read()` da un URL ma non per eseguire un passaggio `write()`.

Utilizzo di Gremlin con il motore di query Neptune DFE

Se si abilita completamente il motore di query alternativo Neptune noto come DFE in modalità di laboratorio (impostando il parametro del cluster database `neptune_lab_mode` su `DFEQueryEngine=enabled`), Neptune converte le query e/o gli attraversamenti Gremlin di sola lettura in una rappresentazione logica intermedia e li esegue sul motore DFE quando possibile.

Tuttavia, il motore DFE non supporta ancora tutti i passaggi Gremlin. Quando un passaggio non può essere eseguito in modo nativo sul DFE, Neptune torna a eseguire il passaggio. TinkerPop I report `explain` e `profile` includono avvisi quando ciò accade.

> **Note**
>
> A partire dal rilascio 1.0.5.0 del motore, il comportamento predefinito del motore DFE per la gestione dei passaggi Gremlin senza supporto nativo è cambiato. Laddove in precedenza il motore DFE ripiegava sul motore Neptune Gremlin, ora ricorre al motore vanilla. TinkerPop

Passaggi Gremlin supportati in modo nativo dal motore DFE

- **`GraphStep`**
- **`VertexStep`**
- **`EdgeVertexStep`**
- **`IdStep`**
- **`TraversalFilterStep`**
- **`PropertiesStep`**
- Supporto del filtro **`HasStep`** per vertici e archi su proprietà, ID ed etichette, ad eccezione di testo e predicati `Without`.
- **`WherePredicateStep`** con filtri con ambito `Path` ma senza supporto della ricerca `ByModulation`, `SideEffect` o `Map`
- **`DedupGlobalStep`**, ad eccezione del supporto della ricerca `ByModulation`, `SideEffect` e `Map`.

Interleaving della pianificazione delle query

Quando il processo di conversione incontra un passaggio Gremlin che non ha un operatore DFE nativo corrispondente, prima di tornare a usare Tinkerpop cerca di trovare altre parti di query intermedie che possano essere eseguite in modo nativo sul motore DFE. Lo fa applicando la logica di interleaving all'attraversamento di livello superiore. Il risultato è che i passaggi supportati vengono utilizzati laddove possibile.

Tale conversione di query intermedia, senza prefisso viene rappresentata utilizzando `NeptuneInterleavingStep` negli output di `explain` e `profile`.

Per confrontare le prestazioni, si potrebbe voler disattivare l'interleaving in una query, continuando a utilizzare il motore DFE per eseguire la parte del prefisso. In alternativa, potreste voler utilizzare solo

il TinkerPop motore per l'esecuzione di query senza prefisso. A tale scopo, utilizzare l'hint di query `disableInterleaving`.

Proprio come l'hint di query useDFE con il valore `false` impedisce del tutto l'esecuzione di una query sul motore DFE, l'hint di query `disableInterleaving` con il valore `true` disattiva l'interleaving del motore DFE per la conversione di una query. Per esempio:

```
g.with('Neptune#disableInterleaving', true)
 .V().has('genre','drama').in('likes')
```

Output aggiornato di **`explain`** e **`profile`** di Gremlin

Gremlin explain fornisce dettagli sull'attraversamento ottimizzato utilizzato da Neptune per eseguire una query. Consultare l'output di esempio di `explain` del motore DFE per un esempio di come appare l'output di `explain` quando il motore DFE è abilitato.

L'API Gremlin `profile` esegue un attraversamento Gremlin specificato, raccoglie varie metriche sull'esecuzione e produce un report di profile che contiene dettagli sul piano di interrogazione ottimizzato e sulle statistiche di runtime di vari operatori. Consultare l'output di esempio di `profile` del motore DFE per un esempio di come appare l'output di `profile` quando il motore DFE è abilitato.

> **Note**
>
> Poiché il motore DFE è una funzionalità sperimentale rilasciata in modalità di laboratorio, il formato esatto dell'output di `explain` e `profile` è soggetto a modifiche.

Accesso al grafo di Neptune con openCypher

Neptune supporta la creazione di applicazioni a grafo mediante openCypher, attualmente uno dei linguaggi di query più popolari per gli sviluppatori che lavorano con database a grafo. Sviluppatori, analisti aziendali e data scientist apprezzano la sintassi ispirata a SQL di openCypher perché fornisce una struttura familiare per comporre query per applicazioni a grafo.

openCypher è un linguaggio di query dichiarativo per grafi di proprietà che è stato sviluppato originariamente da Neo4j, per poi diventare open source nel 2015, e che ha contribuito al progetto openCypher con una licenza open source Apache 2. La sintassi di openCypher è documentata in Cypher Query Language Reference, versione 9.

Per le limitazioni e le differenze nel supporto di Neptune della specifica openCypher, vedi Conformità alle specifiche OpenCypher in Amazon Neptune.

> **Note**
>
> L'attuale implementazione Neo4j del linguaggio di query Cypher si è discostata in qualche modo dalla specifica openCypher. Se si sta eseguendo la migrazione del codice Neo4j Cypher corrente a Neptune, consulta Compatibilità di Neptune con Neo4j e Riscrittura delle query Cypher da eseguire in openCypher su Neptune per ulteriori informazioni.

A partire dal rilascio 1.1.1.0 del motore, openCypher è disponibile per l'uso in produzione in Neptune.

Gremlin e openCypher: somiglianze e differenze

Gremlin e openCypher sono entrambi linguaggi di query per grafi di proprietà e sono complementari in molti modi.

Gremlin è stato progettato per attrarre i programmatori e adattarsi perfettamente al codice. Di conseguenza, Gremlin è fondamentale fin dalla progettazione, mentre la sintassi dichiarativa di openCypher può sembrare più familiare a chi ha esperienza con SQL o SPARQL. Gremlin potrebbe sembrare più naturale per un data scientist che utilizza Python in un notebook Jupyter, mentre openCypher potrebbe sembrare più intuitivo per un utente aziendale con un background SQL.

L'aspetto positivo è che in Neptune non è necessario scegliere tra Gremlin e openCypher. Le query in entrambi i linguaggi possono operare sullo stesso grafo indipendentemente da quale dei due linguaggi sia stato utilizzato per inserire i dati. A seconda di ciò che si fa, può essere più conveniente usare Gremlin per alcune cose e openCypher per altre.

Gremlin utilizza una sintassi imperativa che consente di controllare il modo in cui ci si sposta nel grafo in una serie di passaggi, ognuno dei quali riceve un flusso di dati, esegue alcune azioni su di esso (utilizzando un filtro, una mappa e così via) e quindi invia i risultati al passaggio successivo. Una query Gremlin di solito assume il formato `g.V()`, seguito da passaggi aggiuntivi.

In openCypher si utilizza una sintassi dichiarativa, ispirata a SQL, che specifica un modello di nodi e relazioni da trovare nel grafo utilizzando una sintassi basata su Motif (come `()-[]->()`). Una query openCypher spesso inizia con una clausola `MATCH`, seguita da altre clausole come `WHERE`, `WITH` e `RETURN`.

Nozioni di base sull'uso di openCypher

È possibile eseguire query sui dati del grafo delle proprietà in Neptune utilizzando openCypher indipendentemente da come sono stati caricati, ma non è possibile utilizzare openCypher per eseguire query sui dati caricati come RDF.

Lo strumento di caricamento in blocco Neptune accetta i dati del grafo delle proprietà in formato CSV per Gremlin e in formato CSV per openCypher. Inoltre, ovviamente, è possibile aggiungere dati delle proprietà al grafo usando le query Gremlin e/o openCypher.

Sono disponibili molti tutorial online per imparare il linguaggio di query Cypher. Qui, alcuni rapidi esempi di query openCypher possono aiutarti a farti un'idea del linguaggio, ma di gran lunga il modo migliore e più semplice per iniziare a usare openCypher per eseguire query sul grafo Neptune è usare i notebook openCypher in Neptune Workbench. L'ambiente di lavoro è open source ed è ospitato all'indirizzo https://github.com/aws-samples/. GitHub amazon-neptune-samples

Troverai i taccuini OpenCypher nel repository di taccuini grafici Neptune. GitHub In particolare, consulta i notebook Air-routes visualization e English Premier Teams per openCypher.

I dati elaborati da openCypher assumono la forma di una serie non ordinata di mappe chiave/valore. Il modo principale per perfezionare, manipolare e incrementare queste mappe consiste nell'utilizzare clausole che eseguono attività come la corrispondenza, l'inserimento, l'aggiornamento e l'eliminazione di modelli nelle coppie chiave/valore.

Esistono diverse clausole in openCypher per trovare modelli di dati nel grafo, di cui `MATCH` è la più comune. `MATCH` consente di specificare il modello di nodi, relazioni e filtri che si desidera cercare nel grafo. Per esempio:

- Ottenere tutti i nodi

```
MATCH (n) RETURN n
```

- Trovare i nodi connessi

```
MATCH (n)-[r]->(d) RETURN n, r, d
```

- Trovare un percorso

```
MATCH p=(n)-[r]->(d) RETURN p
```

- Ottenere tutti i nodi con un'etichetta

```
MATCH (n:airport) RETURN n
```

Notare che la prima query restituisce ogni singolo nodo del grafo mentre le due successive restituiscono ogni nodo che ha una relazione: questa operazione non è generalmente consigliata. In quasi tutti i casi, si desidera restringere i dati restituiti, specificando le etichette e le proprietà dei nodi o delle relazioni, come nel quarto esempio.

È possibile trovare un pratico riepilogo di informazioni sulla sintassi di openCypher nel repository di esempi GitHub di Neptune.

Servlet di stato ed endpoint di stato openCypher di Neptune

L'endpoint di stato openCypher fornisce l'accesso alle informazioni sulle query attualmente in esecuzione sul server o in attesa di essere eseguite. Consente inoltre di annullare tali query. L'endpoint è:

```
https://(the server):(the port number)/openCypher/status
```

È possibile utilizzare i metodi HTTP `GET` e `POST` per ottenere lo stato corrente dal server o per annullare una query. È inoltre possibile utilizzare il metodo `DELETE` per annullare una query in esecuzione o in attesa.

Parametri per le richieste di stato

Parametri delle query di stato

- **`includeWaiting`** (`true` o `false`): se impostato su `true` e non sono presenti altri parametri, vengono restituite le informazioni sullo stato per le query in attesa e per quelle in esecuzione.
- **`cancelQuery`**: utilizzato solo con i metodi `GET` e `POST`, per indicare che si tratta di una richiesta di annullamento. Il metodo `DELETE` non richiede questo parametro.

  Il valore del parametro `cancelQuery` non viene utilizzato, ma quando `cancelQuery` è presente, è necessario il parametro `queryId` per identificare la query da annullare.
- **`queryId`**: contiene l'ID di una query specifica.

Se utilizzato con il metodo `GET` o `POST` e il parametro `cancelQuery` non è presente, `queryId` restituisce informazioni sullo stato per la query specifica che identifica. Se il parametro `cancelQuery` è presente, la query specifica identificata da `queryId` viene annullata.

Se utilizzato con il metodo `DELETE`, `queryId` indica sempre una query specifica da annullare.

- **`silent`**: utilizzato solo quando si annulla una query. Se impostato su `true`, l'annullamento avviene senza alcun avviso.

Campi di risposta della richiesta di stato

Campi di risposta di stato se non viene fornito l'ID di una query specifica

- acceptedQueryCount— Il numero di richieste che sono state accettate ma non ancora completate, incluse le domande in coda.
- runningQueryCount— Il numero di query OpenCypher attualmente in esecuzione.
- queries: elenco delle query openCypher correnti.

Campi di risposta di stato per una query specifica

- queryId: ID GUID della query. Neptune assegna automaticamente questo valore ID a ogni query oppure è possibile assegnare un ID personalizzato (consulta Inserimento di un ID personalizzato in una query Neptune Gremlin o SPARQL).
- queryString: la query inviata. Questa è troncata a 1024 caratteri nel caso in cui sia più lunga.
- queryEvalStats— Statistiche per questa interrogazione:
  - waited: indica il tempo di attesa della query, in millisecondi.
  - elapsed: numero di millisecondi in cui la query è stata eseguita finora.
  - cancelled: `True` indica che la query è stata annullata, `False` che non è stata annullata.

Esempi di richieste e risposte di stato

- Richiesta dello stato di tutte le query, comprese quelle in attesa:

```
curl https://server:port/openCypher/status \
  --data-urlencode "includeWaiting=true"
```

Risposta:

```
{
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- Richiesta dello stato delle query in esecuzione, escluse quelle in attesa:

```
curl https://server:port/openCypher/status
```

Risposta:

```
{
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- Richiesta dello stato di una singola query:

```
curl https://server:port/openCypher/status \
 --data-urlencode "queryId=eadc6eea-698b-4a2f-8554-5270ab17ebee"
```

Risposta:

```
{
  "queryId" : "eadc6eea-698b-4a2f-8554-5270ab17ebee",
  "queryString" : "MATCH (n1)-[:knows]->(n2), (n2)-[:knows]->(n3), (n3)-[:knows]-
>(n4), (n4)-[:knows]->(n5), (n5)-[:knows]->(n6), (n6)-[:knows]->(n7), (n7)-[:knows]-
>(n8), (n8)-[:knows]->(n9), (n9)-[:knows]->(n10) RETURN COUNT(n1);",
  "queryEvalStats" : {
    "waited" : 0,
    "elapsed" : 23463,
    "cancelled" : false
  }
}
```

- Richieste di annullamento di una query

1. Con POST:

```
curl -X POST https://server:port/openCypher/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=f43ce17b-db01-4d37-a074-c76d1c26d7a9"
```

Risposta:

```
{
  "status" : "200 OK",
  "payload" : true
}
```

2. Con GET:

```
curl -X GET https://server:port/openCypher/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=588af350-cfde-4222-bee6-b9cedc87180d"
```

Risposta:

```
{
  "status" : "200 OK",
  "payload" : true
}
```

3. Con DELETE:

```
curl -X DELETE \
  -s "https://server:port/openCypher/status?queryId=b9a516d1-d25c-4301-
bb80-10b2743ecf0e"
```

Risposta:

```
{
  "status" : "200 OK",
  "payload" : true
}
```

Endpoint HTTPS openCypher di Amazon Neptune

Argomenti

Query di lettura e scrittura openCypher sull'endpoint HTTPS

L'endpoint HTTPS openCypher supporta le query di lettura e aggiornamento utilizzando i metodi `GET` e `POST`. I metodi `DELETE` e `PUT` non sono supportati.

Le istruzioni seguenti mostrano come connettersi all'endpoint openCypher utilizzando il comando `curl` e HTTPS. Segui queste istruzioni da un'istanza Amazon EC2 nello stesso cloud privato virtuale (VPC) dell'istanza database Neptune.

La sintassi è:

```
HTTPS://(the server):(the port number)/openCypher
```

Ecco alcune query di lettura di esempio, una che utilizza `POST` e una che utilizza `GET`:

1. Con `POST`:

```
curl HTTPS://server:port/openCypher \
  -d "query=MATCH (n1) RETURN n1;"
```

2. Con `GET` (la stringa di query è codificata come URL):

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=MATCH%20(n1)%20RETURN%20n1"
```

Ecco alcune query di scrittura/aggiornamento di esempio, una che utilizza `POST` e una che utilizza `GET`:

1. Con `POST`:

```
curl HTTPS://server:port/openCypher \
  -d "query=CREATE (n:Person { age: 25 })"
```

2. Con GET (la stringa di query è codificata come URL):

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=CREATE%20(n%3APerson%20%7B%20age%3A
%2025%20%7D)"
```

Formato dei risultati JSON predefinito di openCypher

Il seguente formato JSON viene restituito per impostazione predefinita o impostando l'intestazione della richiesta in modo esplicito su `Accept: application/json`. Questo formato è stato progettato per essere facilmente analizzato in oggetti utilizzando le funzionalità del linguaggio nativo della maggior parte delle librerie.

Il documento JSON restituito contiene un campo, `results`, che contiene i valori restituiti dalla query. Gli esempi seguenti mostrano la formattazione JSON per i valori comuni.

Esempio di risposta di valore:

```
{
  "results": [
    {
      "count(a)": 121
    }
  ]
}
```

Esempio di risposta di nodo:

```
{
  "results": [
    {
      "a": {
        "~id": "22",
        "~entityType": "node",
        "~labels": [
          "airport"
        ],
        "~properties": {
          "desc": "Seattle-Tacoma",
          "lon": -122.30899810791,
          "runways": 3,
          "type": "airport",
```

```
          "country": "US",
          "region": "US-WA",
          "lat": 47.4490013122559,
          "elev": 432,
          "city": "Seattle",
          "icao": "KSEA",
          "code": "SEA",
          "longest": 11901
        }
      }
    }
  ]
}
```

Esempio di risposta di relazione:

```
{
  "results": [
    {
      "r": {
        "~id": "7389",
        "~entityType": "relationship",
        "~start": "22",
        "~end": "151",
        "~type": "route",
        "~properties": {
          "dist": 956
        }
      }
    }
  ]
}
```

Esempio di risposta di percorso:

```
{
  "results": [
    {
      "p": [
        {
          "~id": "22",
          "~entityType": "node",
          "~labels": [
```

```
            "airport"
          ],
          "~properties": {
            "desc": "Seattle-Tacoma",
            "lon": -122.30899810791,
            "runways": 3,
            "type": "airport",
            "country": "US",
            "region": "US-WA",
            "lat": 47.4490013122559,
            "elev": 432,
            "city": "Seattle",
            "icao": "KSEA",
            "code": "SEA",
            "longest": 11901
          }
        },
        {
          "~id": "7389",
          "~entityType": "relationship",
          "~start": "22",
          "~end": "151",
          "~type": "route",
          "~properties": {
            "dist": 956
          }
        },
        {
          "~id": "151",
          "~entityType": "node",
          "~labels": [
            "airport"
          ],
          "~properties": {
            "desc": "Ontario International Airport",
            "lon": -117.600997924805,
            "runways": 2,
            "type": "airport",
            "country": "US",
            "region": "US-CA",
            "lat": 34.0559997558594,
            "elev": 944,
            "city": "Ontario",
            "icao": "KONT",
```

```
            "code": "ONT",
            "longest": 12198
          }
        }
      ]
    }
  ]
}
```

Utilizzo del protocollo Bolt per effettuare query openCypher in Neptune

Bolt è un protocollo client/server orientato alle istruzioni inizialmente sviluppato da Neo4j e concesso in licenza con la licenza Creative Commons 3.0 Attribution-. ShareAlike È basato sul client, il che significa che il client avvia sempre lo scambio di messaggi.

Per connettersi a Neptune utilizzando i driver Bolt di Neo4j, è sufficiente sostituire l'URL e il numero di porta con gli endpoint del cluster utilizzando lo schema URI `bolt`. Se è in esecuzione una singola istanza di Neptune, usare l'endpoint read_write. Se sono in esecuzione più istanze, si consigliano due driver, uno per la scrittura e l'altro per tutte le repliche di lettura. Se si dispone solo dei due endpoint predefiniti, sono sufficienti un driver read_write e uno read_only, ma se si dispone anche di endpoint personalizzati, valutare la possibilità di creare un'istanza del driver per ciascuno di essi.

> **Note**
>
> Sebbene le specifiche Bolt affermino che Bolt può connettersi tramite TCP o WebSockets, Neptune supporta solo connessioni TCP per Bolt.

Neptune consente fino a 1.000 connessioni Bolt simultanee.

Per esempi di query openCypher in vari linguaggi che utilizzano i driver Bolt, consulta la documentazione Drivers & Language Guides di Neo4j.

> **Important**
>
> I driver Neo4j Bolt per Python JavaScript, .NET e Golang inizialmente non supportavano il rinnovo automatico dei token di autenticazione Signature v4. AWS Ciò significa che dopo la scadenza della firma (spesso dopo 5 minuti), il driver non riusciva ad autenticarsi e le richieste successive avevano esito negativo. Gli esempi Python JavaScript, .NET e Go riportati di seguito sono stati tutti interessati da questo problema.

Per ulteriori informazioni, vedere Neo4j Python numero #834, Neo4j .NET numero #664, driver Neo4j numero #993 e JavaScript driver Neo4j GoLang numero #429.
A partire dalla versione 5.8.0 del driver, è stata rilasciata una nuova API di riautenticazione in anteprima per il driver Go (vedi v5.8.0 - Feedback wanted on re-authentication).

Utilizzo di Bolt con Java per connettersi a Neptune

È possibile scaricare un driver per la versione che si desidera utilizzare dal repository MVN Maven o aggiungere questa dipendenza al progetto:

```
<dependency>
  <groupId>org.neo4j.driver</groupId>
  <artifactId>neo4j-java-driver</artifactId>
  <version>4.3.3</version>
</dependency>
```

Quindi, per connettersi a Neptune in Java utilizzando uno di questi driver Bolt, creare un'istanza del driver per l'istanza primaria/di scrittura nel cluster usando un codice simile al seguente:

```
import org.neo4j.driver.Driver;
import org.neo4j.driver.GraphDatabase;

final Driver driver =
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
    AuthTokens.none(),
    Config.builder().withEncryption()
                    .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                    .build());
```

Se si dispone di una o più repliche di lettura, è possibile creare in modo analogo un'istanza del driver per tali repliche utilizzando un codice simile al seguente:

```
final Driver read_only_driver =              // (without connection timeout)
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
      Config.builder().withEncryption()
                      .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                      .build());
```

Oppure, con un timeout:

```
final Driver read_only_timeout_driver =      // (with connection timeout)
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
    Config.builder().withConnectionTimeout(30, TimeUnit.SECONDS)
                    .withEncryption()
                    .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                    .build());
```

Se si dispone di endpoint personalizzati, può anche essere utile creare un'istanza del driver per ognuno di essi.

Esempio di query openCypher in Python con Bolt

Ecco come creare una query openCypher in Python con Bolt:

```
python -m pip install neo4j
```

```
from neo4j import GraphDatabase
uri = "bolt://(your cluster endpoint URL):(your cluster port)"
driver = GraphDatabase.driver(uri, auth=("username", "password"), encrypted=True)
```

Notare che i parametri `auth` vengono ignorati.

Esempio di query openCypher in .NET con Bolt

Per creare una query OpenCypher in .NET utilizzando Bolt, il primo passo è installare il driver Neo4j utilizzando. NuHet Per effettuare chiamate sincrone, usare la versione `.Simple`, come in questo caso:

```
Install-Package Neo4j.Driver.Simple-4.3.0
```

```
using Neo4j.Driver;

namespace hello
{
  // This example creates a node and reads a node in a Neptune
  // Cluster where IAM Authentication is not enabled.
  public class HelloWorldExample : IDisposable
  {
    private bool _disposed = false;
    private readonly IDriver _driver;
```

```
   private static string url = "bolt://(your cluster endpoint URL):(your cluster
port)";
   private static string createNodeQuery = "CREATE (a:Greeting) SET a.message =
'HelloWorldExample'";
   private static string readNodeQuery = "MATCH(n:Greeting) RETURN n.message";

   ~HelloWorldExample() => Dispose(false);

   public HelloWorldExample(string uri)
   {
     _driver = GraphDatabase.Driver(uri, AuthTokens.None, o =>
o.WithEncryptionLevel(EncryptionLevel.Encrypted));
   }

   public void createNode()
   {
     // Open a session
     using (var session = _driver.Session())
     {
        // Run the query in a write transaction
       var greeting = session.WriteTransaction(tx =>
       {
         var result = tx.Run(createNodeQuery);
         // Consume the result
         return result.Consume();
       });

       // The output will look like this:
       //   ResultSummary{Query=`CREATE (a:Greeting) SET a.message =
'HelloWorldExample".....
       Console.WriteLine(greeting);
     }
   }

   public void retrieveNode()
   {
     // Open a session
     using (var session = _driver.Session())
     {
       // Run the query in a read transaction
       var greeting = session.ReadTransaction(tx =>
       {
         var result = tx.Run(readNodeQuery);
         // Consume the result. Read the single node
```

```
          // created in a previous step.
          return result.Single()[0].As<string>();
        });
        // The output will look like this:
        //   HelloWorldExample
        Console.WriteLine(greeting);
      }
    }

    public void Dispose()
    {
      Dispose(true);
      GC.SuppressFinalize(this);
    }

    protected virtual void Dispose(bool disposing)
    {
      if (_disposed)
        return;
      if (disposing)
      {
        _driver?.Dispose();
      }
      _disposed = true;
    }

    public static void Main()
    {
      using (var apiCaller = new HelloWorldExample(url))
      {
        apiCaller.createNode();
        apiCaller.retrieveNode();
      }
    }
  }
}
```

Esempio di query openCypher in Java con Bolt e autenticazione IAM

Il codice Java riportato di seguito mostra come effettuare query openCypher in Java con Bolt e autenticazione IAM. Il commento JavaDoc ne descrive l'utilizzo. Una volta disponibile un'istanza del driver, è possibile utilizzarla per effettuare più richieste autenticate.

```
package software.amazon.neptune.bolt;

import com.amazonaws.DefaultRequest;
import com.amazonaws.Request;
import com.amazonaws.auth.AWS4Signer;
import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.http.HttpMethodName;
import com.google.gson.Gson;
import lombok.Builder;
import lombok.Getter;
import lombok.NonNull;
import org.neo4j.driver.Value;
import org.neo4j.driver.Values;
import org.neo4j.driver.internal.security.InternalAuthToken;
import org.neo4j.driver.internal.value.StringValue;

import java.net.URI;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;

import static com.amazonaws.auth.internal.SignerConstants.AUTHORIZATION;
import static com.amazonaws.auth.internal.SignerConstants.HOST;
import static com.amazonaws.auth.internal.SignerConstants.X_AMZ_DATE;
import static com.amazonaws.auth.internal.SignerConstants.X_AMZ_SECURITY_TOKEN;

/**
 * Use this class instead of `AuthTokens.basic` when working with an IAM
 * auth-enabled server. It works the same as `AuthTokens.basic` when using
 * static credentials, and avoids making requests with an expired signature
 * when using temporary credentials. Internally, it generates a new signature
 * on every invocation (this may change in a future implementation).
 *
 * Note that authentication happens only the first time for a pooled connection.
 *
 * Typical usage:
 *
 * NeptuneAuthToken authToken = NeptuneAuthToken.builder()
 *     .credentialsProvider(credentialsProvider)
 *     .region("aws region")
 *     .url("cluster endpoint url")
 *     .build();
 *
```

```
 * Driver driver = GraphDatabase.driver(
 *     authToken.getUrl(),
 *     authToken,
 *     config
 * );
 */

public class NeptuneAuthToken extends InternalAuthToken {
  private static final String SCHEME = "basic";
  private static final String REALM = "realm";
  private static final String SERVICE_NAME = "neptune-db";
  private static final String HTTP_METHOD_HDR = "HttpMethod";
  private static final String DUMMY_USERNAME = "username";
  @NonNull
  private final String region;
  @NonNull
  @Getter
  private final String url;
  @NonNull
  private final AWSCredentialsProvider credentialsProvider;
  private final Gson gson = new Gson();

  @Builder
  private NeptuneAuthToken(
      @NonNull final String region,
      @NonNull final String url,
      @NonNull final AWSCredentialsProvider credentialsProvider
  ) {
      // The superclass caches the result of toMap(), which we don't want
      super(Collections.emptyMap());
      this.region = region;
      this.url = url;
      this.credentialsProvider = credentialsProvider;
  }

  @Override
  public Map<String, Value> toMap() {
    final Map<String, Value> map = new HashMap<>();
    map.put(SCHEME_KEY, Values.value(SCHEME));
    map.put(PRINCIPAL_KEY, Values.value(DUMMY_USERNAME));
    map.put(CREDENTIALS_KEY, new StringValue(getSignedHeader()));
    map.put(REALM_KEY, Values.value(REALM));

    return map;
```

```
  }

  private String getSignedHeader() {
    final Request<Void> request = new DefaultRequest<>(SERVICE_NAME);
    request.setHttpMethod(HttpMethodName.GET);
    request.setEndpoint(URI.create(url));
    // Comment out the following line if you're using an engine version older than
 1.2.0.0
    request.setResourcePath("/opencypher");

    final AWS4Signer signer = new AWS4Signer();
    signer.setRegionName(region);
    signer.setServiceName(request.getServiceName());
    signer.sign(request, credentialsProvider.getCredentials());

    return getAuthInfoJson(request);
  }

  private String getAuthInfoJson(final Request<Void> request) {
    final Map<String, Object> obj = new HashMap<>();
    obj.put(AUTHORIZATION, request.getHeaders().get(AUTHORIZATION));
    obj.put(HTTP_METHOD_HDR, request.getHttpMethod());
    obj.put(X_AMZ_DATE, request.getHeaders().get(X_AMZ_DATE));
    obj.put(HOST, request.getHeaders().get(HOST));
    obj.put(X_AMZ_SECURITY_TOKEN, request.getHeaders().get(X_AMZ_SECURITY_TOKEN));

    return gson.toJson(obj);
  }
}
```

Esempio di query openCypher in Python con Bolt e autenticazione IAM

La classe Python seguente consente di effettuare query openCypher in Python con Bolt e autenticazione IAM:

```
import json

from neo4j import Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import Credentials
from botocore.auth import (
  SigV4Auth,
  _host_from_url,
```

```
)

SCHEME = "basic"
REALM = "realm"
SERVICE_NAME = "neptune-db"
DUMMY_USERNAME = "username"
HTTP_METHOD_HDR = "HttpMethod"
HTTP_METHOD = "GET"
AUTHORIZATION = "Authorization"
X_AMZ_DATE = "X-Amz-Date"
X_AMZ_SECURITY_TOKEN = "X-Amz-Security-Token"
HOST = "Host"


class NeptuneAuthToken(Auth):
  def __init__(
    self,
    credentials: Credentials,
    region: str,
    url: str,
    **parameters
  ):
    # Do NOT add "/opencypher" in the line below if you're using an engine version
 older than 1.2.0.0
    request = AWSRequest(method=HTTP_METHOD, url=url + "/opencypher")
    request.headers.add_header("Host", _host_from_url(request.url))
    sigv4 = SigV4Auth(credentials, SERVICE_NAME, region)
    sigv4.add_auth(request)

    auth_obj = {
      hdr: request.headers[hdr]
      for hdr in [AUTHORIZATION, X_AMZ_DATE, X_AMZ_SECURITY_TOKEN, HOST]
    }
    auth_obj[HTTP_METHOD_HDR] = request.method
    creds: str = json.dumps(auth_obj)
    super().__init__(SCHEME, DUMMY_USERNAME, creds, REALM, **parameters)
```

Utilizzare questa classe per creare un driver come segue:

```
  authToken = NeptuneAuthToken(creds, REGION, URL)
  driver = GraphDatabase.driver(URL, auth=authToken, encrypted=True)
```

Esempio di Node.js con autenticazione IAM e Bolt

Il codice Node.js riportato di seguito utilizza la sintassi AWS SDK per la JavaScript versione 3 ed ES6 per creare un driver che autentica le richieste:

```
import neo4j from "neo4j-driver";
import { HttpRequest }  from "@aws-sdk/protocol-http";
import { defaultProvider } from "@aws-sdk/credential-provider-node";
import { SignatureV4 } from "@aws-sdk/signature-v4";
import crypto from "@aws-crypto/sha256-js";
const { Sha256 } = crypto;
import assert from "node:assert";

const region = "us-west-2";
const serviceName = "neptune-db";
const host = "(your cluster endpoint URL)";
const port = 8182;
const protocol = "bolt";
const hostPort = host + ":" + port;
const url = protocol + "://" + hostPort;
const createQuery = "CREATE (n:Greeting {message: 'Hello'}) RETURN ID(n)";
const readQuery = "MATCH(n:Greeting) WHERE ID(n) = $id RETURN n.message";

async function signedHeader() {
  const req = new HttpRequest({
    method: "GET",
    protocol: protocol,
    hostname: host,
    port: port,
    // Comment out the following line if you're using an engine version older than
 1.2.0.0
    path: "/opencypher",
    headers: {
      host: hostPort
    }
  });

  const signer = new SignatureV4({
    credentials: defaultProvider(),
    region: region,
    service: serviceName,
    sha256: Sha256
  });
```

```
  return signer.sign(req, { unsignableHeaders: new Set(["x-amz-content-sha256"]) })
    .then((signedRequest) => {
      const authInfo = {
        "Authorization": signedRequest.headers["authorization"],
        "HttpMethod": signedRequest.method,
        "X-Amz-Date": signedRequest.headers["x-amz-date"],
        "Host": signedRequest.headers["host"],
        "X-Amz-Security-Token": signedRequest.headers["x-amz-security-token"]
      };
      return JSON.stringify(authInfo);
    });
}

async function createDriver() {
  let authToken = { scheme: "basic", realm: "realm", principal: "username",
 credentials: await signedHeader() };

  return neo4j.driver(url, authToken, {
      encrypted: "ENCRYPTION_ON",
      trust: "TRUST_SYSTEM_CA_SIGNED_CERTIFICATES",
      maxConnectionPoolSize: 1,
      // logging: neo4j.logging.console("debug")
    }
  );
}

function unmanagedTxn(driver) {
  const session = driver.session();
  const tx = session.beginTransaction();
  tx.run(createQuery)
  .then((res) => {
    const id = res.records[0].get(0);
    return tx.run(readQuery, { id: id });
  })
  .then((res) => {
    // All good, the transaction will be committed
    const msg = res.records[0].get("n.message");
    assert.equal(msg, "Hello");
  })
  .catch(err => {
    // The transaction will be rolled back, now handle the error.
    console.log(err);
  })
```

```
  .then(() => session.close());
}

createDriver()
.then((driver) => {
  unmanagedTxn(driver);
  driver.close();
})
.catch((err) => {
  console.log(err);
});
```

Esempio di query openCypher in .NET con Bolt e autenticazione IAM

Per abilitare l'autenticazione IAM in .NET, è necessario firmare una richiesta quando si stabilisce la connessione. L'esempio seguente mostra come creare un helper `NeptuneAuthToken` per generare un token di autenticazione:

```
using Amazon.Runtime;
using Amazon.Util;
using Neo4j.Driver;
using System.Security.Cryptography;
using System.Text;
using System.Text.Json;
using System.Web;

namespace Hello
{
  /*
   * Use this class instead of `AuthTokens.None` when working with an IAM-auth-enabled
 server.
   *
   * Note that authentication happens only the first time for a pooled connection.
   *
   * Typical usage:
   *
   * var authToken = new NeptuneAuthToken(AccessKey, SecretKey,
 Region).GetAuthToken(Host);
   * _driver = GraphDatabase.Driver(Url, authToken, o =>
 o.WithEncryptionLevel(EncryptionLevel.Encrypted));
   */

  public class NeptuneAuthToken
```

```
 {
   private const string ServiceName = "neptune-db";
   private const string Scheme = "basic";
   private const string Realm = "realm";
   private const string DummyUserName = "username";
   private const string Algorithm = "AWS4-HMAC-SHA256";
   private const string AWSRequest = "aws4_request";

   private readonly string _accessKey;
   private readonly string _secretKey;
   private readonly string _region;

   private readonly string _emptyPayloadHash;

   private readonly SHA256 _sha256;


   public NeptuneAuthToken(string awsKey = null, string secretKey = null, string
region = null)
   {
     var awsCredentials = awsKey == null || secretKey == null
       ? FallbackCredentialsFactory.GetCredentials().GetCredentials()
       : null;

     _accessKey = awsKey ?? awsCredentials.AccessKey;
     _secretKey = secretKey ?? awsCredentials.SecretKey;
     _region = region ?? FallbackRegionFactory.GetRegionEndpoint().SystemName; //ex:
us-east-1

     _sha256 = SHA256.Create();
     _emptyPayloadHash = Hash(Array.Empty<byte>());
   }

   public IAuthToken GetAuthToken(string url)
   {
     return AuthTokens.Custom(DummyUserName, GetCredentials(url), Realm, Scheme);
   }

   /******************** AWS SIGNING FUNCTIONS ********************/
   private string Hash(byte[] bytesToHash)
   {
     return ToHexString(_sha256.ComputeHash(bytesToHash));
   }
```

```
    private static byte[] HmacSHA256(byte[] key, string data)
    {
      return new HMACSHA256(key).ComputeHash(Encoding.UTF8.GetBytes(data));
    }

    private byte[] GetSignatureKey(string dateStamp)
    {
      var kSecret = Encoding.UTF8.GetBytes($"AWS4{_secretKey}");
      var kDate = HmacSHA256(kSecret, dateStamp);
      var kRegion = HmacSHA256(kDate, _region);
      var kService = HmacSHA256(kRegion, ServiceName);
      return HmacSHA256(kService, AWSRequest);
    }

    private static string ToHexString(byte[] array)
    {
      return Convert.ToHexString(array).ToLowerInvariant();
    }

    private string GetCredentials(string url)
    {
      var request = new HttpRequestMessage
      {
        Method = HttpMethod.Get,
        RequestUri = new Uri($"https://{url}/opencypher")
      };

      var signedrequest = Sign(request);

      var headers = new Dictionary<string, object>
      {
        [HeaderKeys.AuthorizationHeader] =
signedrequest.Headers.GetValues(HeaderKeys.AuthorizationHeader).FirstOrDefault(),
        ["HttpMethod"] = HttpMethod.Get.ToString(),
        [HeaderKeys.XAmzDateHeader] =
signedrequest.Headers.GetValues(HeaderKeys.XAmzDateHeader).FirstOrDefault(),
        // Host should be capitalized, not like in Amazon.Util.HeaderKeys.HostHeader
        ["Host"] =
signedrequest.Headers.GetValues(HeaderKeys.HostHeader).FirstOrDefault(),
      };

      return JsonSerializer.Serialize(headers);
    }
```

```
    private HttpRequestMessage Sign(HttpRequestMessage request)
    {
      var now = DateTimeOffset.UtcNow;
      var amzdate = now.ToString("yyyyMMddTHHmmssZ");
      var datestamp = now.ToString("yyyyMMdd");

      if (request.Headers.Host == null)
      {
        request.Headers.Host = $"{request.RequestUri.Host}:{request.RequestUri.Port}";
      }

      request.Headers.Add(HeaderKeys.XAmzDateHeader, amzdate);

      var canonicalQueryParams = GetCanonicalQueryParams(request);

      var canonicalRequest = new StringBuilder();
      canonicalRequest.Append(request.Method + "\n");
      canonicalRequest.Append(request.RequestUri.AbsolutePath + "\n");
      canonicalRequest.Append(canonicalQueryParams + "\n");

      var signedHeadersList = new List<string>();
      foreach (var header in request.Headers.OrderBy(a => a.Key.ToLowerInvariant()))
      {
        canonicalRequest.Append(header.Key.ToLowerInvariant());
        canonicalRequest.Append(':');
        canonicalRequest.Append(string.Join(",", header.Value.Select(s => s.Trim())));
        canonicalRequest.Append('\n');
        signedHeadersList.Add(header.Key.ToLowerInvariant());
      }
      canonicalRequest.Append('\n');

      var signedHeaders = string.Join(";", signedHeadersList);
      canonicalRequest.Append(signedHeaders + "\n");
      canonicalRequest.Append(_emptyPayloadHash);

      var credentialScope = $"{datestamp}/{_region}/{ServiceName}/{AWSRequest}";
      var stringToSign = $"{Algorithm}\n{amzdate}\n{credentialScope}\n"
        + Hash(Encoding.UTF8.GetBytes(canonicalRequest.ToString()));

      var signing_key = GetSignatureKey(datestamp);
      var signature = ToHexString(HmacSHA256(signing_key, stringToSign));

      request.Headers.TryAddWithoutValidation(HeaderKeys.AuthorizationHeader,
```

```
        $"{Algorithm} Credential={_accessKey}/{credentialScope},
 SignedHeaders={signedHeaders}, Signature={signature}");

      return request;
    }

    private static string GetCanonicalQueryParams(HttpRequestMessage request)
    {
      var querystring = HttpUtility.ParseQueryString(request.RequestUri.Query);

      // Query params must be escaped in upper case (i.e. "%2C", not "%2c").
      var queryParams = querystring.AllKeys.OrderBy(a => a)
        .Select(key => $"{key}={Uri.EscapeDataString(querystring[key])}");
      return string.Join("&", queryParams);
    }
  }
}
```

Ecco come creare una query openCypher in .NET con Bolt e autenticazione IAM. L'esempio seguente utilizza l'helper `NeptuneAuthToken`:

```
using Neo4j.Driver;

namespace Hello
{
  public class HelloWorldExample
  {
    private const string Host = "(your hostname):8182";
    private const string Url = $"bolt://{Host}";
    private const string CreateNodeQuery = "CREATE (a:Greeting) SET a.message =
 'HelloWorldExample'";
    private const string ReadNodeQuery = "MATCH(n:Greeting) RETURN n.message";

    private const string AccessKey = "(your access key)";
    private const string SecretKey = "(your secret key)";
    private const string Region = "(your AWS region)"; // e.g. "us-west-2"

    private readonly IDriver _driver;

    public HelloWorldExample()
    {
      var authToken = new NeptuneAuthToken(AccessKey, SecretKey,
 Region).GetAuthToken(Host);
```

```
     // Note that when the connection is reinitialized after max connection lifetime
     // has been reached, the signature token could have already been expired (usually
5 min)
     // You can face exceptions like:
     //   `Unexpected server exception 'Signature expired: XXXX is now earlier than
YYYY (ZZZZ - 5 min.)`
     _driver = GraphDatabase.Driver(Url, authToken, o =>

o.WithMaxConnectionLifetime(TimeSpan.FromMinutes(60)).WithEncryptionLevel(EncryptionLevel.Encr
   }

   public async Task CreateNode()
   {
     // Open a session
     using (var session = _driver.AsyncSession())
     {
       // Run the query in a write transaction
       var greeting = await session.WriteTransactionAsync(async tx =>
       {
         var result = await tx.RunAsync(CreateNodeQuery);
         // Consume the result
         return await result.ConsumeAsync();
       });

       // The output will look like this:
       //   ResultSummary{Query=`CREATE (a:Greeting) SET a.message =
'HelloWorldExample".....
       Console.WriteLine(greeting.Query);
     }
   }

   public async Task RetrieveNode()
   {
     // Open a session
     using (var session = _driver.AsyncSession())
     {
       // Run the query in a read transaction
       var greeting = await session.ReadTransactionAsync(async tx =>
       {
         var result = await tx.RunAsync(ReadNodeQuery);
         var records = await result.ToListAsync();

         // Consume the result. Read the single node
```

```
          // created in a previous step.
          return records[0].Values.First().Value;
        });
        // The output will look like this:
        //   HelloWorldExample
        Console.WriteLine(greeting);
      }
    }
  }
}
```

Questo esempio può essere avviato eseguendo il codice sottostante su `.NET 6` o `.NET 7` con i seguenti pacchetti:

- **`Neo4j`**`.Driver=4.3.0`
- **`AWSSDK`**`.Core=3.7.102.1`

```
namespace Hello
{
  class Program
  {
    static async Task Main()
    {
      var apiCaller = new HelloWorldExample();

      await apiCaller.CreateNode();
      await apiCaller.RetrieveNode();
    }
  }
}
```

Esempio di query openCypher in Golang con Bolt e autenticazione IAM

Il pacchetto Golang seguente mostra come effettuare query openCypher nel linguaggio Go con Bolt e autenticazione IAM:

```
package main

import (
  "context"
  "encoding/json"
```

```
  "fmt"
  "github.com/aws/aws-sdk-go/aws/credentials"
  "github.com/aws/aws-sdk-go/aws/signer/v4"
  "github.com/neo4j/neo4j-go-driver/v5/neo4j"
  "log"
  "net/http"
  "os"
  "time"
)

const (
  ServiceName   = "neptune-db"
  DummyUsername = "username"
)

// Find node by id using Go driver
func findNode(ctx context.Context, region string, hostAndPort string, nodeId string)
 (string, error) {
  req, err := http.NewRequest(http.MethodGet, "https://"+hostAndPort+"/opencypher",
 nil)

  if err != nil {
    return "", fmt.Errorf("error creating request, %v", err)
  }

  // credentials must have been exported as environment variables
  signer := v4.NewSigner(credentials.NewEnvCredentials())
  _, err = signer.Sign(req, nil, ServiceName, region, time.Now())

  if err != nil {
    return "", fmt.Errorf("error signing request: %v", err)
  }

  hdrs := []string{"Authorization", "X-Amz-Date", "X-Amz-Security-Token"}
  hdrMap := make(map[string]string)
  for _, h := range hdrs {
    hdrMap[h] = req.Header.Get(h)
  }

  hdrMap["Host"] = req.Host
  hdrMap["HttpMethod"] = req.Method

  password, err := json.Marshal(hdrMap)
  if err != nil {
```

```
    return "", fmt.Errorf("error creating JSON, %v", err)
  }
  authToken := neo4j.BasicAuth(DummyUsername, string(password), "")
  // +s enables encryption with a full certificate check
  // Use +ssc to disable client side TLS verification
  driver, err := neo4j.NewDriverWithContext("bolt+s://"+hostAndPort+"/opencypher",
 authToken)
  if err != nil {
    return "", fmt.Errorf("error creating driver, %v", err)
  }

  defer driver.Close(ctx)

  if err := driver.VerifyConnectivity(ctx); err != nil {
    log.Fatalf("failed to verify connection, %v", err)
  }

  config := neo4j.SessionConfig{}

  session := driver.NewSession(ctx, config)
  defer session.Close(ctx)

  result, err := session.Run(
    ctx,
    fmt.Sprintf("MATCH (n) WHERE ID(n) = '%s' RETURN n", nodeId),
    map[string]any{},
  )
  if err != nil {
    return "", fmt.Errorf("error running query, %v", err)
  }

  if !result.Next(ctx) {
    return "", fmt.Errorf("node not found")
  }

  n, found := result.Record().Get("n")
  if !found {
    return "", fmt.Errorf("node not found")
  }

  return fmt.Sprintf("+%v\n", n), nil
}

func main() {
```

```
  if len(os.Args) < 3 {
    log.Fatal("Usage: go main.go (region) (host and port)")
  }
  region := os.Args[1]
  hostAndPort := os.Args[2]
  ctx := context.Background()

  res, err := findNode(ctx, region, hostAndPort,
 "72c2e8c1-7d5f-5f30-10ca-9d2bb8c4afbc")
  if err != nil {
    log.Fatal(err)
  }
  fmt.Println(res)
}
```

Comportamento della connessione Bolt in Neptune

Di seguito sono elencati alcuni punti da tenere a mente sulle connessioni Bolt in Neptune:

- Poiché le connessioni Bolt vengono create a livello TCP, non è possibile utilizzare un Application Load Balancer con tali connessioni, come è possibile fare con un endpoint HTTP.
- La porta utilizzata da Neptune per le connessioni Bolt è la porta del cluster database.
- In base al preambolo Bolt che gli è stato passato, il server Neptune seleziona la versione Bolt più elevata appropriata (1, 2, 3 o 4.0).
- Il numero massimo di connessioni al server Neptune che un client può avere aperte in qualsiasi momento è 1.000.
- Se il client non chiude una connessione dopo una query, tale connessione può essere utilizzata per eseguire la query successiva.
- Tuttavia, se una connessione rimane inattiva per 20 minuti, il server la chiude automaticamente.
- Se l'autenticazione IAM non è abilitata, è possibile utilizzare `AuthTokens.none()` anziché fornire un nome utente e una password fittizi. Ad esempio, in Java:

  ```
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
   AuthTokens.none(),

   Config.builder().withEncryption().withTrustStrategy(TrustStrategy.trustSystemCertificates())
  ```

- Quando l'autenticazione IAM è abilitata, una connessione Bolt viene sempre disconnessa 10 giorni e alcuni minuti dopo essere stata stabilita, se non è già stata chiusa per qualche altro motivo.

- Se il client invia una query per l'esecuzione tramite una connessione senza aver utilizzato i risultati di una query precedente, la nuova query viene eliminata. Per eliminare invece i risultati precedenti, il client deve inviare un messaggio di ripristino tramite la connessione.
- Su una determinata connessione è possibile creare una sola transazione alla volta.
- Se si verifica un'eccezione durante una transazione, il server Neptune esegue il rollback della transazione e chiude la connessione. In questo caso, il driver crea una nuova connessione per la query successiva.
- Tenere presente che le sessioni non sono thread-safe. Più operazioni parallele devono utilizzare più sessioni separate.

Esempi di query openCypher parametrizzate

Neptune supporta query openCypher parametrizzate. Ciò consente di utilizzare la stessa struttura di query più volte con argomenti diversi. Poiché la struttura della query non cambia, Neptune può memorizzare nella cache l'albero sintattico astratto (AST) invece di doverlo analizzare più volte.

Esempio di una query openCypher parametrizzata con l'endpoint HTTPS

Di seguito è riportato un esempio di utilizzo di una query parametrizzata con l'endpoint HTTPS openCypher di Neptune. La query è:

```
MATCH (n {name: $name, age: $age})
RETURN n
```

I parametri sono definiti come segue:

```
parameters={"name": "john", "age": 20}
```

Con `GET` è possibile inviare la query parametrizzata in questo modo:

```
curl -k \
  "https://localhost:8182/openCypher?query=MATCH%20%28n%20%7Bname:\$name,age:\$age%7D
%29%20RETURN%20n&parameters=%7B%22name%22:%22john%22,%22age%22:20%7D"
```

In alternativa, è possibile utilizzare `POST`:

```
curl -k \
  https://localhost:8182/openCypher \
```

```
  -d "query=MATCH (n {name: \$name, age: \$age}) RETURN n" \
  -d "parameters={\"name\": \"john\", \"age\": 20}"
```

Oppure, con `DIRECT POST`:

```
curl -k \
   -H "Content-Type: application/opencypher" \
  "https://localhost:8182/openCypher?parameters=%7B%22name%22:%22john%22,%22age
%22:20%7D" \
  -d "MATCH (n {name: \$name, age: \$age}) RETURN n"
```

Esempi di query openCypher parametrizzate con Bolt

Ecco un esempio Python di una query openCypher parametrizzata che utilizza il protocollo Bolt:

```
from neo4j import GraphDatabase
uri = "bolt://[neptune-endpoint-url]:8182"
driver = GraphDatabase.driver(uri, auth=("", ""))

def match_name_and_age(tx, name, age):
  # Parameterized Query
  tx.run("MATCH (n {name: $name, age: $age}) RETURN n", name=name, age=age)

with driver.session() as session:
  # Parameters
  session.read_transaction(match_name_and_age, "john", 20)

driver.close()
```

Ecco un esempio Java di una query openCypher parametrizzata che utilizza il protocollo Bolt:

```
Driver driver = GraphDatabase.driver("bolt+s://(your cluster endpoint URL):8182");
HashMap<String, Object> parameters = new HashMap<>();
parameters.put("name", "john");
parameters.put("age", 20);
String queryString = "MATCH (n {name: $name, age: $age}) RETURN n";
Result result = driver.session().run(queryString, parameters);
```

Modello di dati openCypher

Il motore openCypher di Neptune si basa sullo stesso modello di grafo delle proprietà di Gremlin. In particolare:

- Ogni nodo ha una o più etichette. Se si inserisce un nodo senza etichette, viene associata un'etichetta predefinita denominata `vertex`. Se si tenta di eliminare tutte le etichette di un nodo, viene generato un errore.
- Una relazione è un'entità che ha esattamente un tipo di relazione e che forma una connessione unidirezionale tra due nodi (ovvero da uno dei nodi all'altro).
- Sia i nodi che le relazioni possono avere proprietà, ma non è obbligatorio. Neptune supporta nodi e relazioni con zero proprietà.
- Neptune non supporta le metaproprietà, che non sono incluse nemmeno nelle specifiche openCypher.
- Le proprietà del grafo possono contenere più valori se sono state create con Gremlin. In altre parole, la proprietà di un nodo o di una relazione può avere un insieme di valori diversi anziché uno solo. Neptune ha esteso la semantica di openCypher per gestire correttamente le proprietà multivalore.

I tipi di dati supportati sono documentati in Formato dati openCypher. Tuttavia, al momento non è consigliabile inserire i valori della proprietà `Array` in un grafo openCypher. Sebbene sia possibile inserire un valore della proprietà array utilizzando lo strumento di caricamento in blocco, il rilascio corrente openCypher di Neptune lo tratta come un insieme di proprietà multivalore anziché come un singolo valore di elenco.

Di seguito è riportato l'elenco dei tipi di dati supportati in questo rilascio:

- `Bool`
- `Byte`
- `Short`
- `Int`
- `Long`
- `Float` (include +/- Infinity e NaN, ma non INF)
- `Double` (include +/- Infinity e NaN, ma non INF)
- `DateTime`
- `String`

Funzionalità **explain** di openCypher

La funzionalità explain di openCypher è uno strumento self-service di Amazon Neptune che consente di comprendere l'approccio di esecuzione adottato dal motore Neptune. Per richiamare explain, si passa un parametro a una richiesta HTTPS openCypher con `explain=`*`mode`*, dove il valore *mode* può essere uno dei seguenti:

- **`static`**: in modalità `static`, `explain` visualizza solo la struttura statica del piano di query. Non esegue effettivamente la query.
- **`dynamic`**: in modalità `dynamic`, `explain` esegue anche la query e include gli aspetti dinamici del piano di query. Questi aspetti potrebbero includere il numero di associazioni intermedie che passano attraverso gli operatori, il rapporto tra le associazioni in entrata e quelle in uscita e il tempo totale impiegato da ogni operatore.
- **`details`**: in modalità `details`, `explain` visualizza le informazioni mostrate in modalità dynamic oltre a dettagli aggiuntivi, come la stringa di query openCypher effettiva e il calcolo dell'intervallo stimato per il modello sottostante un operatore join.

Ad esempio, con POST:

```
curl HTTPS://server:port/openCypher \
  -d "query=MATCH (n) RETURN n LIMIT 1;" \
  -d "explain=dynamic"
```

Oppure, con GET:

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=MATCH%20(n)%20RETURN%20n%20LIMIT
%201&explain=dynamic"
```

Limitazioni per **explain** di openCypher in Neptune

Il rilascio corrente di explain di openCypher presenta le seguenti limitazioni:

- I piani di explain sono attualmente disponibili solo per le query che eseguono operazioni di sola lettura. Le query che eseguono qualsiasi tipo di mutazione, come `CREATE`, `DELETE`, `MERGE`, `SET` e così via non sono supportate.

- Gli operatori e l'output di un piano specifico potrebbero cambiare nei rilasci futuri.

Operatori DFE nell'output di **`explain`** di openCypher

Per utilizzare le informazioni fornite dalla funzionalità `explain` di openCypher, è necessario comprendere alcuni dettagli su come funziona il motore di query DFE (DFE è il motore utilizzato da Neptune per elaborare le query openCypher).

Il motore DFE traduce ogni query SPARQL in una pipeline di operatori. Partendo dal primo operatore, le soluzioni intermedie passano da un operatore all'altro attraverso questa pipeline di operatori. Ogni riga della tabella di explain rappresenta un risultato, fino al punto di valutazione.

Gli operatori che possono essere presenti in un piano di query DFE sono i seguenti:

DFEApply: esegue la funzione specificata nella sezione degli argomenti, sul valore archiviato nella variabile specificata

DFE: associa le variabili con i BindRelation nomi specificati

DFE ChunkLocalSubQuery — Si tratta di un'operazione non bloccante che funge da involucro per l'esecuzione delle sottoquery.

DFE DistinctColumn — Restituisce il sottoinsieme distinto dei valori di input in base alla variabile specificata.

DFE DistinctRelation — Restituisce il sottoinsieme distinto delle soluzioni di input in base alla variabile specificata.

DFEDrain: viene visualizzato alla fine di una sottoquery e funge da passaggio di terminazione per la sottoquery. Il numero di soluzioni viene registrato come `Units In`. `Units Out` è sempre zero.

DFE ForwardValue — Copia tutti i blocchi di input direttamente come blocchi di output da passare al relativo operatore a valle.

DFE GroupByHashIndex — Esegue un'operazione di raggruppamento sulle soluzioni di input in base a un indice hash calcolato in precedenza (utilizzando l'operazione). `DFEHashIndexBuild` Analogamente a un output, l'input specificato viene esteso di una colonna contenente una chiave di gruppo per ogni soluzione di input.

DFE HashIndexBuild — Crea un indice hash su un insieme di variabili come effetto collaterale. Questo indice hash viene in genere riutilizzato nelle operazioni successive. Consulta

`DFEHashIndexJoin` o `DFEGroupByHashIndex` per informazione su dove potrebbe essere necessario utilizzare questo indice.

DFE HashIndexJoin: esegue un join sulle soluzioni in entrata rispetto a un indice hash creato in precedenza. Consulta `DFEHashIndexBuild` per informazioni su dove potrebbe essere necessario creare questo indice.

DFE JoinExists: accetta una relazione di input sinistra e destra e conserva i valori della relazione sinistra che hanno un valore corrispondente nella relazione destra, come definito dalle variabili di join fornite.

: si tratta di un'operazione non bloccante che funge da wrapper per una sottoquery, consentendone l'esecuzione ripetuta per l'utilizzo nei cicli.

DFE MergeChunks: si tratta di un'operazione di blocco che combina i blocchi dell'operatore a monte in un unico blocco di soluzioni da passare all'operatore a valle (inversa di). `DFESplitChunks`

DFEMinus: accetta una relazione di input sinistra e destra e mantiene i valori della relazione sinistra che non hanno un valore corrispondente nella relazione destra, come definito dalle variabili di join specificate. Se non vi è alcuna sovrapposizione nelle variabili tra entrambe le relazioni, questo operatore restituisce semplicemente la relazione di input sinistra.

DFE NotExists: accetta una relazione di input sinistra e destra e mantiene i valori della relazione sinistra che non hanno un valore corrispondente nella relazione destra, come definito dalle variabili di join fornite. Se non vi è alcuna sovrapposizione nelle variabili in entrambe le relazioni, questo operatore restituisce una relazione vuota.

DFE OptionalJoin — Esegue un join esterno sinistro (chiamato anche join OPZIONALE): le soluzioni dal lato sinistro che hanno almeno un partner di unione sul lato destro vengono unite e le soluzioni dal lato sinistro senza partner di unione sul lato destro vengono inoltrate così come sono. Questa è un'operazione bloccante.

DFE PipelineJoin: unisce l'input al modello di tuple definito dall'argomento. `pattern`

DFE PipelineRangeCount: conta il numero di soluzioni che corrispondono a un determinato modello e restituisce una singola soluzione monoaria contenente il valore di conteggio.

DFE PipelineScan: esegue la scansione del database per l'`pattern`argomento specificato, con o senza un determinato filtro sulle colonne.

DFEProject: accetta più colonne di input e proietta solo le colonne desiderate.

DFEReduce: esegue la funzione di aggregazione specificata su variabili specificate.

DFE RelationalJoin — Unisce l'input dell'operatore precedente in base alle chiavi del pattern specificato utilizzando un merge join. Questa è un'operazione bloccante.

DFE RouteChunks: preleva i blocchi di input dal suo singolo bordo in entrata e li indirizza lungo i suoi molteplici bordi in uscita.

DFE SelectRows: questo operatore preleva selettivamente le righe dalle sue soluzioni di relazione di input sinistro per inoltrarle all'operatore a valle. Le righe selezionate in base agli identificatori di riga forniti nella relazione di input destra dell'operatore.

DFESerialize: serializza i risultati finali di una query in una serializzazione di stringhe JSON, associando ogni soluzione di input al nome di variabile appropriato. Per i risultati di nodi ed edge, questi risultati vengono serializzati in una mappa delle proprietà e dei metadati delle entità.

DFESort: accetta una relazione di input e produce una relazione ordinata basata sulla chiave di ordinamento fornita.

DFE SplitByGroup — Divide ogni singolo blocco di input da un bordo di ingresso in blocchi di output più piccoli corrispondenti ai gruppi di righe identificati dagli ID di riga del blocco di input corrispondente dall'altro bordo in ingresso.

DFE SplitChunks — Divide ogni singolo blocco di input in blocchi di output più piccoli (inversa di). `DFEMergeChunks`

DFE — Versione in streaming di. StreamingHashIndexBuild `DFEHashIndexBuild`

DFE StreamingGroupByHashIndex - Versione in streaming di. `DFEGroupByHashIndex`

DFESubquery: questo operatore appare all'inizio di tutti i piani e incapsula le parti del piano che vengono eseguite sul motore DFE, che è l'intero piano per openCypher.

DFE SymmetricHashJoin: unisce l'input dell'operatore precedente in base alle chiavi del pattern specificate utilizzando un hash join. Questa è un'operazione non bloccante.

DFESync: è un operatore di sincronizzazione che supporta piani non bloccanti. Accetta le soluzioni da due edge in entrata e le inoltra agli edge a valle appropriati. Ai fini della sincronizzazione, gli input lungo uno di questi edge possono essere bufferizzati internamente.

DFETee: si tratta di un operatore di ramificazione che invia lo stesso set di soluzioni a più operatori.

DFE TermResolution: esegue un'operazione di localizzazione o globalizzazione sui relativi input, generando colonne rispettivamente di identificatori localizzati o globalizzati.

: espande gli elenchi di valori da una colonna di input nella colonna di output come singoli elementi.

DFEUnion: accetta due o più relazioni di input e produce un'unione di tali relazioni utilizzando lo schema di output desiderato.

SolutionInjection— Viene visualizzato prima di ogni altra cosa nell'output di spiegazione, con un valore di 1 nella colonna Units Out. Tuttavia, non genera alcuna operazione e in realtà non inserisce alcuna soluzione nel motore DFE.

TermResolution— Appare alla fine dei piani e traduce gli oggetti dal motore Neptune in oggetti OpenCypher.

Colonne nell'output di **`explain`** di openCypher

Le informazioni sul piano di query che Neptune genera come output di explain di openCypher contengono tabelle con un operatore per riga. La tabella contiene le seguenti colonne:

ID: ID numerico di questo operatore nel piano.

Out #1 (e Out #2): ID degli operatori downstream rispetto a questo operatore. Possono esserci al massimo due operatori downstream.

Name: nome di questo operatore.

Arguments: qualsiasi dettaglio rilevante per l'operatore. Ciò include elementi come lo schema di input, lo schema di output, il modello (per `PipelineScan` e `PipelineJoin`) e così via.

Mode: etichetta che descrive il comportamento fondamentale dell'operatore. Questa colonna è per lo più vuota (`-`). Un'eccezione è `TermResolution`, dove mode può essere `id2value_opencypher`, che indica una risoluzione dall'ID al valore openCypher.

Units In: numero di soluzioni passate come input a questo operatore. Gli operatori senza operatori upstream, come `DFEPipelineScan`, `SolutionInjections` e `DFESubquery` senza valore statico inserito, avranno valore zero.

Units Out: numero di soluzioni prodotte come output di questo operatore. `DFEDrain` è un caso speciale, in cui il numero di soluzioni da eliminare viene registrato in `Units In` e `Units Out` è sempre zero.

Ratio: rapporto tra `Units Out` e `Units In`.

Time (ms): tempo della CPU utilizzato da questo operatore, in millisecondi.

Esempio di base dell'output di explain di openCypher

Il seguente è un esempio di base dell'output di `explain` di openCypher. La query è una ricerca a nodo singolo nel set di dati delle rotte aeree per un nodo con il codice aeroportuale `ATL` che richiama `explain` usando la modalità `details` nel formato di output ASCII predefinito:

```
curl -d "query=MATCH (n {code: 'ATL'}) RETURN n" -k https://localhost:8182/openCypher -
d "explain=details"
                                    ~
Query:
MATCH (n {code: 'ATL'}) RETURN n

#################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
#################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 1         # 0.00  # 4.00      #
#################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?n]          # id2value_opencypher #
 1        # 1         # 1.00  # 2.00      #
#################################################################################################


subQuery1
#################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?n) with property 'code'
 as ?n_code2 and label 'ALL'                                      # -    # 0
   # 1         # 0.00  # 0.21      #
#    #        #        #                      # inlineFilters=[(?n_code2 IN
 ["ATL"^^xsd:string])]                                                  #
  #          #           #       #           #
```

```
#    #        #        #                      # patternEstimate=1
                                                                  #      #
  #           #       #           #
#########################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#9d84f97c-c3b0-459a-98d5-955a8726b159/graph_1 # -    #
 1        # 1         # 1.00  # 0.04      #
#########################################################################################
# 2  # 3      # -      # DFEProject            # columns=[?n]
                                                                  # -    # 1
  # 1         # 1.00  # 0.04      #
#########################################################################################
# 3  # -      # -      # DFEDrain              # -
                                                                  # -    # 1
  # 0         # 0.00  # 0.03      #
#########################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#9d84f97c-
c3b0-459a-98d5-955a8726b159/graph_1
#########################################################################################
# ID # Out #1 # Out #2 # Name                 # Arguments
                     # Mode # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?n, ?n_code2]
                     # -    # 0        # 1         # 0.00  # 0.02      #
#########################################################################################
# 1  # 2      # 3      # DFETee               # -
                     # -    # 1        # 2         # 2.00  # 0.02      #
#########################################################################################
# 2  # 4      # -      # DFEDistinctColumn    # column=?n
                     # -    # 1        # 1         # 1.00  # 0.20      #
#    #        #        #                      # ordered=false
                     #      #          #           #       #           #
#########################################################################################
# 3  # 5      # -      # DFEHashIndexBuild    # vars=[?n]
                     # -    # 1        # 1         # 1.00  # 0.04      #
#########################################################################################
# 4  # 5      # -      # DFEPipelineJoin      # pattern=Node(?n) with property 'ALL'
 and label '?n_label1' # -    # 1        # 1         # 1.00  # 0.25      #
#    #        #        #                      # patternEstimate=3506
                     #      #          #           #       #           #
#########################################################################################
```

```
# 5  # 6      # 7      # DFESync              # -
                    # -    # 2        # 2         # 1.00  # 0.02      #
#########################################################################################
# 6  # 8      # -      # DFEForwardValue      # -
                    # -    # 1        # 1         # 1.00  # 0.01      #
#########################################################################################
# 7  # 8      # -      # DFEForwardValue      # -
                    # -    # 1        # 1         # 1.00  # 0.01      #
#########################################################################################
# 8  # 9      # -      # DFEHashIndexJoin     # -
                    # -    # 2        # 1         # 0.50  # 0.35      #
#########################################################################################
# 9  # -      # -      # DFEDrain              # -
                    # -    # 1        # 0         # 0.00  # 0.02      #
#########################################################################################
```

Al livello superiore, `SolutionInjection` appare prima di ogni altra cosa, con 1 come valore di Units In. Notare che in realtà non inserisce alcuna soluzione. Si può vedere che l'operatore successivo `DFESubquery` ha 0 come valore di Units Out.

Dopo `SolutionInjection` al livello più alto, ci sono gli operatori `DFESubquery` e `TermResolution`. `DFESubquery` incapsula le parti del piano di esecuzione delle query che viene inviato al motore DFE (per le query openCypher, l'intero piano di esecuzione delle query viene eseguito dal motore DFE). Tutti gli operatori del piano di query sono annidati all'interno di `subQuery1` a cui fa riferimento `DFESubquery`. L'unica eccezione è `TermResolution`, che materializza gli ID interni in oggetti openCypher completamente serializzati.

Tutti gli operatori che vengono inviati al motore DFE hanno nomi che iniziano con un prefisso `DFE`. Come accennato in precedenza, l'intero piano di query di openCypher viene eseguito dal motore DFE, quindi, tutti gli operatori tranne l'operatore finale `TermResolution` iniziano con `DFE`.

All'interno di `subQuery1` possono essere presenti zero o più operatori `DFEChunkLocalSubQuery` o `DFELoopSubQuery` che incapsulano una parte del piano di esecuzione inviato che viene eseguito in un meccanismo con limiti di memoria. `DFEChunkLocalSubQuery` qui contiene un solo `SolutionInjection` che viene utilizzato come input per la sottoquery. Per trovare la tabella per tale sottoquery nell'output, cercare `subQuery=`*`graph URI`* specificato nella colonna `Arguments` per l'operatore `DFEChunkLocalSubQuery` o `DFELoopSubQuery`.

In `subQuery1`, `DFEPipelineScan` con `ID` 0 analizza il database alla ricerca di un oggetto `pattern` specificato. Il modello cerca un'entità con proprietà `code` salvata come variabile `?n_code2` su tutte

le etichette (è possibile filtrare in base a un'etichetta specifica aggiungendo `airport` a `n:airport`). L'argomento `inlineFilters` mostra il filtro per la proprietà `code` uguale a `ATL`.

Successivamente, l'operatore `DFEChunkLocalSubQuery` esegue il join dei risultati intermedi di una sottoquery che contiene `DFEPipelineJoin`. Ciò garantisce che `?n` sia effettivamente un nodo, poiché l'operatore precedente `DFEPipelineScan` cerca qualsiasi entità con la proprietà `code`.

Esempio di output di **`explain`** per una ricerca di relazioni con un limite

Questa query cerca le relazioni tra due nodi anonimi con tipo `route` e ne restituisce al massimo 10. Anche in questo caso, la modalità `explain` è `details` e il formato di output è il formato ASCII predefinito. Ecco l'output di `explain`:

Qui, `DFEPipelineScan` cerca gli archi che iniziano dal nodo anonimo `?anon_node7` e terminano in un altro nodo anonimo `?anon_node21`, con un tipo di relazione salvato come `?p_type1`. Esiste un filtro per `?p_type1` che è `el://route` (dove `el` indica l'etichetta dell'arco), che corrisponde a `[p:route]` nella stringa di query.

`DFEDrain` raccoglie la soluzione di output con un limite di 10, come mostrato nella relativa colonna `Arguments`. `DFEDrain` termina una volta raggiunto il limite o quando sono state prodotte tutte le soluzioni, a seconda dell'evento che si verifica per primo.

```
curl -d "query=MATCH ()-[p:route]->() RETURN p LIMIT 10" -k https://localhost:8182/
openCypher -d "explain=details"
                                        ~
Query:
MATCH ()-[p:route]->() RETURN p LIMIT 10

#################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
#################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 10        # 0.00  # 5.00      #
#################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?p]          # id2value_opencypher #
 10       # 10        # 1.00  # 1.00      #
#################################################################################################
```

```
subQuery1
#################################################################################################
# ID # Out #1 # Out #2 # Name            # Arguments
              # Mode # Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # DFEPipelineScan # pattern=Edge((?anon_node7)-[?p:?p_type1]->(?
anon_node21)) # -    # 0        # 1000      # 0.00  # 0.66      #
#    #        #        #                 # inlineFilters=[(?p_type1 IN [<el://route>])]
              #      #          #           #       #           #
#    #        #        #                 # patternEstimate=26219
              #      #          #           #       #           #
#################################################################################################
# 1  # 2      # -      # DFEProject      # columns=[?p]
              # -    # 1000     # 1000      # 1.00  # 0.14      #
#################################################################################################
# 2  # -      # -      # DFEDrain        # limit=10
              # -    # 1000     # 0         # 0.00  # 0.11      #
#################################################################################################
```

Esempio di output di **`explain`** per una funzione di espressione di valori

La funzione è:

```
MATCH (a) RETURN DISTINCT labels(a)
```

Nell'output di `explain` seguente, `DFEPipelineScan` (ID 0) cerca tutte le etichette dei nodi. Ciò corrisponde a `MATCH (a)`.

`DFEChunkLocalSubquery` (ID 1) aggrega l'etichetta di `?a` per ogni `?a`. Ciò corrisponde a `labels(a)`. Si può vederlo attraverso `DFEApply` e `DFEReduce`.

`BindRelation` (ID 2) viene utilizzato per rinominare la colonna generica `?__gen_labelsOfa2` in `?labels(a)`.

`DFEDistinctRelation` (ID 4) recupera solo le etichette distinte (più nodi :airport restituiranno valori labels(a) duplicati: ["airport"]). Ciò corrisponde a `DISTINCT labels(a)`.

```
curl -d "query=MATCH (a) RETURN DISTINCT labels(a)" -k https://localhost:8182/
openCypher -d "explain=details"
                                          ~
Query:
```

```
MATCH (a) RETURN DISTINCT labels(a)

################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 5         # 0.00  # 81.00     #
################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?labels(a)]  # id2value_opencypher #
 5        # 5         # 1.00  # 1.00      #
################################################################################################


subQuery1
################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?a) with property 'ALL'
 and label '?a_label1'                                              # -    # 0
    # 3750      # 0.00  # 26.77     #
#    #        #        #                       # patternEstimate=3506
                                                                 #      #
  #           #       #           #
################################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#8b314f55-2cc7-456a-a48a-c76a0465cfab/graph_1 # -    #
 3750     # 3750      # 1.00  # 0.04      #
################################################################################################
# 2  # 3      # -      # DFEBindRelation       # inputVars=[?a, ?__gen_labelsOfa2, ?
__gen_labelsOfa2]                                                   # -    # 3750
     # 3750      # 1.00  # 0.08      #
#    #        #        #                       # outputVars=[?a, ?__gen_labelsOfa2, ?
labels(a)]                                                          #      #
    #           #       #           #
################################################################################################
# 3  # 4      # -      # DFEProject            # columns=[?labels(a)]
                                                                  # -    # 3750
  # 3750      # 1.00  # 0.05      #
```

```
#######################################################################################################
# 4  # 5      # -      # DFEDistinctRelation   # -
                                                                                  # -    # 3750
  # 5         # 0.00  # 2.78      #
#######################################################################################################
# 5  # -      # -      # DFEDrain              # -
                                                                                  # -    # 5
  # 0         # 0.00  # 0.03      #
#######################################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#8b314f55-2cc7-456a-
a48a-c76a0465cfab/graph_1
#######################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                     # Mode     # Units In # Units Out # Ratio # Time (ms) #
#######################################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?a]
                     # -        # 0        # 3750      # 0.00  # 0.02      #
#######################################################################################################
# 1  # 2      # 3      # DFETee               # -
                     # -        # 3750     # 7500      # 2.00  # 0.02      #
#######################################################################################################
# 2  # 4      # -      # DFEProject           # columns=[?a]
                     # -        # 3750     # 3750      # 1.00  # 0.04      #
#######################################################################################################
# 3  # 17     # -      # DFEOptionalJoin      # -
                     # -        # 7500     # 3750      # 0.50  # 0.44      #
#######################################################################################################
# 4  # 5      # -      # DFEDistinctRelation  # -
                     # -        # 3750     # 3750      # 1.00  # 2.23      #
#######################################################################################################
# 5  # 6      # -      # DFEDistinctColumn    # column=?a
                     # -        # 3750     # 3750      # 1.00  # 1.50      #
#    #        #        #                      # ordered=false
                     #          #          #           #       #           #
#######################################################################################################
# 6  # 7      # -      # DFEPipelineJoin      # pattern=Node(?a) with property 'ALL'
 and label '?a_label3' # -        # 3750     # 3750      # 1.00  # 10.58     #
#    #        #        #                      # patternEstimate=3506
                     #          #          #           #       #           #
#######################################################################################################
# 7  # 8      # 9      # DFETee               # -
                     # -        # 3750     # 7500      # 2.00  # 0.02      #
```

```
#########################################################################################
# 8  # 10     # -      # DFEBindRelation      # inputVars=[?a_label3]
                    # -        # 3750     # 3750      # 1.00  # 0.04      #
#    #        #        #                     # outputVars=[?100]
                    #          #          #           #       #           #
#########################################################################################
# 9  # 11     # -      # DFEBindRelation      # inputVars=[?a, ?a_label3, ?100]
                    # -        # 7500     # 3750      # 0.50  # 0.07      #
#    #        #        #                     # outputVars=[?a, ?a_label3, ?100]
                    #          #          #           #       #           #
#########################################################################################
# 10 # 9      # -      # DFETermResolution    # column=?100
                    # id2value # 3750     # 3750      # 1.00  # 7.60      #
#########################################################################################
# 11 # 12     # -      # DFEBindRelation      # inputVars=[?a, ?a_label3, ?100]
                    # -        # 3750     # 3750      # 1.00  # 0.06      #
#    #        #        #                     # outputVars=[?a, ?100, ?a_label3]
                    #          #          #           #       #           #
#########################################################################################
# 12 # 13     # -      # DFEApply             # functor=nodeLabel(?a_label3)
                    # -        # 3750     # 3750      # 1.00  # 0.55      #
#########################################################################################
# 13 # 14     # -      # DFEProject           # columns=[?a, ?a_label3_alias4]
                    # -        # 3750     # 3750      # 1.00  # 0.05      #
#########################################################################################
# 14 # 15     # -      # DFEMergeChunks       # -
                    # -        # 3750     # 3750      # 1.00  # 0.02      #
#########################################################################################
# 15 # 16     # -      # DFEReduce            # functor=collect(?a_label3_alias4)
                    # -        # 3750     # 3750      # 1.00  # 6.37      #
#    #        #        #                     # segmentationKey=[?a]
                    #          #          #           #       #           #
#########################################################################################
# 16 # 3      # -      # DFEMergeChunks       # -
                    # -        # 3750     # 3750      # 1.00  # 0.03      #
#########################################################################################
# 17 # -      # -      # DFEDrain             # -
                    # -        # 3750     # 0         # 0.00  # 0.02      #
#########################################################################################
```

Esempio di output di **`explain`** per una funzione di espressione di valori matematici

In questo esempio, `RETURN abs(-10)` esegue una valutazione semplice, accettando il valore assoluto di una costante `-10`.

`DFEChunkLocalSubQuery` (ID 1) esegue l'inserimento di una soluzione per il valore statico `-10`, che viene archiviato nella variabile `?100`.

`DFEApply` (ID 2) è l'operatore che esegue la funzione di valore assoluto `abs()` sul valore statico archiviato nella variabile `?100`.

Ecco la query e l'output di `explain` risultante:

```
curl -d "query=RETURN abs(-10)" -k https://localhost:8182/openCypher  -d
 "explain=details"
                                                    ~
Query:
RETURN abs(-10)

##########################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments             # Mode
  # Units In # Units Out # Ratio # Time (ms) #
##########################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]        # -
  # 0        # 1        # 0.00  # 0         #
##########################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1    # -
  # 0        # 1        # 0.00  # 4.00      #
##########################################################################################
# 2  # -      # -      # TermResolution    # vars=[?_internalVar1] #
 id2value_opencypher # 1        # 1         # 1.00  # 1.00      #
##########################################################################################


subQuery1
##########################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
##########################################################################################
# 0  # 1      # -      # DFESolutionInjection  # outSchema=[]
                                                                # -    # 0
  # 1         # 0.00  # 0.01      #
```

```
#########################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#c4cc6148-cce3-4561-93c0-deb91f257356/graph_1 # -    #
 1        # 1         # 1.00  # 0.03      #
#########################################################################################
# 2  # 3      # -      # DFEApply              # functor=abs(?100)
                                                                          # -    # 1
  # 1         # 1.00  # 0.26      #
#########################################################################################
# 3  # 4      # -      # DFEBindRelation       # inputVars=[?_internalVar2, ?
_internalVar2]                                                             # -
 # 1        # 1         # 1.00  # 0.04      #
#    #        #        #                       # outputVars=[?_internalVar2, ?
_internalVar1]                                                           #
 #          #          #       #           #
#########################################################################################
# 4  # 5      # -      # DFEProject            # columns=[?_internalVar1]
                                                                          # -    # 1
  # 1         # 1.00  # 0.06      #
#########################################################################################
# 5  # -      # -      # DFEDrain              # -
                                                                          # -    # 1
  # 0         # 0.00  # 0.05      #
#########################################################################################

subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#c4cc6148-
cce3-4561-93c0-deb91f257356/graph_1
#########################################################################################
# ID # Out #1 # Out #2 # Name                 # Arguments                          #
 Mode # Units In # Units Out # Ratio # Time (ms) #
#########################################################################################
# 0  # 1      # -      # DFESolutionInjection # solutions=[?100 -> [-10^^<LONG>]]   # -
    # 0        # 1         # 0.00  # 0.01      #
#    #        #        #                       # outSchema=[?100]                   #
    #          #          #       #           #
#########################################################################################
# 1  # 3      # -      # DFERelationalJoin    # joinVars=[]                        # -
    # 2        # 1         # 0.50  # 0.18      #
#########################################################################################
# 2  # 1      # -      # DFESolutionInjection # outSchema=[]                      # -
    # 0        # 1         # 0.00  # 0.01      #
#########################################################################################
# 3  # -      # -      # DFEDrain              # -                                  # -
    # 1        # 0         # 0.00  # 0.02      #
```

```
################################################################################################
```

Esempio di output di **`explain`** per una query con percorso a lunghezza variabile (VLP)

Questo è un esempio di un piano di query più complesso per la gestione di una query con percorso a lunghezza variabile. Questo esempio mostra solo una parte dell'output di `explain`, per motivi di chiarezza.

In `subQuery1`, `DFEPipelineScan` (ID 0) e `DFEChunkLocalSubQuery` (ID 1), che inserisce la sottoquery `...graph_1`, sono responsabili della ricerca di un nodo con il codice YP0.

In `subQuery1`, `DFEChunkLocalSubQuery` (ID 2), che inserisce la sottoquery `...graph_2`, è responsabile della ricerca di un nodo con il codice LAX.

In`subQuery1`, `DFEChunkLocalSubQuery` (ID 3) inserisce la sottoquery `...graph3`, che contiene `DFELoopSubQuery` (ID 17), che a sua volta inserisce la sottoquery `...graph5`. Questa operazione è responsabile della risoluzione del modello a lunghezza variabile `-[*2]->` nella stringa di query tra due nodi.

```
curl -d "query=MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p" -k https://
localhost:8182/openCypher -d "explain=details"

                                                    ~
Query:
MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p

################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -       # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
################################################################################################
# 1  # 2      # -       # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 0         # 0.00  # 84.00     #
################################################################################################
# 2  # -      # -       # TermResolution    # vars=[?p]          # id2value_opencypher #
 0        # 0         # 0.00  # 0         #
################################################################################################


subQuery1
```

```
################################################################################################
# ID # Out #1 # Out #2 # Name                    # Arguments
                                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFEPipelineScan        # pattern=Node(?a) with property 'code'
 as ?a_code7 and label 'ALL'                                                       # -    # 0
   # 1         # 0.00  # 0.68      #
#    #        #        #                        # inlineFilters=[(?a_code7 IN
 ["YPO"^^xsd:string])]                                                                  #
  #          #           #       #           #
#    #        #        #                        # patternEstimate=1
                                                                                 #      #
  #           #       #           #
################################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_1 # -    #
 1        # 1         # 1.00  # 0.03      #
################################################################################################
# 2  # 3      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_2 # -    #
 1        # 1         # 1.00  # 0.02      #
################################################################################################
# 3  # 4      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_3 # -    #
 1        # 0         # 0.00  # 0.04      #
################################################################################################
# 4  # 5      # -      # DFEBindRelation       # inputVars=[?__gen_path6, ?
anon_rel26, ?b_code8, ?b, ?a_code7, ?a, ?__gen_path6]                              # -
   # 0        # 0         # 0.00  # 0.10      #
#    #        #        #                        # outputVars=[?__gen_path6, ?
anon_rel26, ?b_code8, ?b, ?a_code7, ?a, ?p]                                        #
  #          #           #       #           #
################################################################################################
# 5  # 6      # -      # DFEProject            # columns=[?p]
                                                                                  # -    # 0
  # 0         # 0.00  # 0.05      #
################################################################################################
# 6  # -      # -      # DFEDrain              # -
                                                                                  # -    # 0
  # 0         # 0.00  # 0.02      #
################################################################################################
```

```
subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_1
#################################################################################################
# ID # Out #1 # Out #2 # Name                 # Arguments
                       # Mode # Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?a, ?a_code7]
                       # -    # 0        # 1         # 0.00  # 0.01      #
#################################################################################################
# 1  # 2      # 3      # DFETee               # -
                       # -    # 1        # 2         # 2.00  # 0.01      #
#################################################################################################
# 2  # 4      # -      # DFEDistinctColumn    # column=?a
                       # -    # 1        # 1         # 1.00  # 0.25      #
#    #        #        #                      # ordered=false
                       #      #          #           #       #           #
#################################################################################################
# 3  # 5      # -      # DFEHashIndexBuild    # vars=[?a]
                       # -    # 1        # 1         # 1.00  # 0.05      #
#################################################################################################
# 4  # 5      # -      # DFEPipelineJoin      # pattern=Node(?a) with property 'ALL'
 and label '?a_label1' # -    # 1        # 1         # 1.00  # 0.47      #
#    #        #        #                      # patternEstimate=3506
                       #      #          #           #       #           #
#################################################################################################
# 5  # 6      # 7      # DFESync              # -
                       # -    # 2        # 2         # 1.00  # 0.04      #
#################################################################################################
# 6  # 8      # -      # DFEForwardValue      # -
                       # -    # 1        # 1         # 1.00  # 0.01      #
#################################################################################################
# 7  # 8      # -      # DFEForwardValue      # -
                       # -    # 1        # 1         # 1.00  # 0.01      #
#################################################################################################
# 8  # 9      # -      # DFEHashIndexJoin     # -
                       # -    # 2        # 1         # 0.50  # 0.26      #
#################################################################################################
# 9  # -      # -      # DFEDrain             # -
                       # -    # 1        # 0         # 0.00  # 0.02      #
#################################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_2
```

```
################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                           # Mode # Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFEPipelineScan      # pattern=Node(?b) with property 'code'
 as ?b_code8 and label 'ALL' # -    # 0        # 1         # 0.00  # 0.38      #
#    #        #        #                      # inlineFilters=[(?b_code8 IN
 ["LAX"^^xsd:string])]                #      #          #           #       #
  #
#    #        #        #                      # patternEstimate=1
                           #      #          #           #       #           #
################################################################################################
# 1  # 2      # -      # DFEMergeChunks       # -
                           # -    # 1        # 1         # 1.00  # 0.02      #
################################################################################################
# 2  # 4      # -      # DFERelationalJoin    # joinVars=[]
                           # -    # 2        # 1         # 0.50  # 0.19      #
################################################################################################
# 3  # 2      # -      # DFESolutionInjection # outSchema=[?a, ?a_code7]
                           # -    # 0        # 1         # 0.00  # 0         #
################################################################################################
# 4  # -      # -      # DFEDrain             # -
                           # -    # 1        # 0         # 0.00  # 0.01      #
################################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_3
################################################################################################
# ID # Out #1 # Out #2 # Name                   # Arguments
                                                                    # Mode     #
 Units In # Units Out # Ratio # Time (ms) #
################################################################################################
...
# 17 # 18     # -      # DFELoopSubQuery       # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_5 # -
  # 1        # 2         # 2.00  # 0.31      #
...
```

Transazioni in Neptune openCypher

L'implementazione openCypher in Amazon Neptune utilizza la semantica delle transazioni definita da Neptune. Tuttavia, i livelli di isolamento forniti dal driver Bolt hanno alcune implicazioni specifiche per la semantica delle transazioni Bolt, come descritto nelle sezioni seguenti.

Query sulle transazioni Bolt di sola lettura

Esistono vari modi in cui è possibile elaborare le query di sola lettura, con diversi modelli di transazione e livelli di isolamento, come segue:

Query sulle transazioni implicite di sola lettura

Di seguito è illustrato un esempio di transazione implicita di sola lettura:

```
public void executeReadImplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
          Config.builder().withEncryption()
                          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                          .build());

  // create the session config
  SessionConfig sessionConfig = SessionConfig.builder()
                                             .withFetchSize(1000)
                                             .withDefaultAccessMode(AccessMode.READ)
                                             .build();

  // run the query as access mode read
  driver.session(sessionConfig).readTransaction(new TransactionWork<String>()
    {
      final StringBuilder resultCollector = new StringBuilder();

      @Override
      public String execute(final Transaction tx)
```

```
      {
        // execute the query
        Result queryResult = tx.run(READ_QUERY);

        // Read the result
        for (Record record : queryResult.list())
        {
          for (String key : record.keys())
          {
            resultCollector.append(key)
                           .append(":")
                           .append(record.get(key).asNode().toString());
          }
        }
        return resultCollector.toString();
      }

    }
  );

  // close the driver.
  driver.close();
}
```

Poiché le repliche di lettura accettano solo query di sola lettura, tutte le query relative sulle repliche di lettura vengono eseguite come transazioni implicite di lettura indipendentemente dalla modalità di accesso impostata nella configurazione della sessione. Neptune valuta le transazioni implicite di lettura come query di sola lettura in base alla semantica di isolamento SNAPSHOT.

In caso di errore, le transazioni implicite di lettura vengono ripetute per impostazione predefinita.

Query sulle transazioni di sola lettura con commit automatico

Di seguito è illustrato un esempio di transazione di sola lettura con commit automatico:

```
public void executeAutoCommitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";
```

```
  // Create the session config.
  final SessionConfig sessionConfig = SessionConfig
    .builder()
    .withFetchSize(1000)
    .withDefaultAccessMode(AccessMode.READ)
    .build();

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
    Config.builder()
          .withEncryption()
          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
          .build());

  // result collector
  final StringBuilder resultCollector = new StringBuilder();

  // create a session
  final Session session = driver.session(sessionConfig);

  // run the query
  final Result queryResult = session.run(READ_QUERY);
  for (final Record record : queryResult.list())
  {
    for (String key : record.keys())
    {
      resultCollector.append(key)
                     .append(":")
                     .append(record.get(key).asNode().toString());
    }
  }

  // close the session
  session.close();

  // close the driver
  driver.close();
}
```

Se la modalità di accesso è impostata su `READ` nella configurazione della sessione, Neptune valuta le query sulle transazioni con commit automatico come query di sola lettura in base alla semantica di isolamento `SNAPSHOT`. Notare che le repliche di lettura accettano solo query di sola lettura.

Se non si esegue una configurazione di sessione, le query con commit automatico vengono elaborate per impostazione predefinita con l'isolamento delle query di mutazione, quindi è importante eseguire una configurazione di sessione che imposti esplicitamente la modalità di accesso su `READ`.

In caso di errore, le query di sola lettura con commit automatico non vengono ripetute.

Query sulle transazioni esplicite di sola lettura

Di seguito è illustrato un esempio di transazione esplicita di sola lettura:

```
public void executeReadExplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";

  // Create the session config.
  final SessionConfig sessionConfig = SessionConfig
    .builder()
    .withFetchSize(1000)
    .withDefaultAccessMode(AccessMode.READ)
    .build();

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
    Config.builder()
          .withEncryption()
          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
          .build());

  // result collector
  final StringBuilder resultCollector = new StringBuilder();

  // create a session
  final Session session = driver.session(sessionConfig);

  // begin transaction
  final Transaction tx = session.beginTransaction();

  // run the query on transaction
  final List<Record> list = tx.run(READ_QUERY).list();
```

```
  // read the result
  for (final Record record : list)
  {
    for (String key : record.keys())
    {
      resultCollector
        .append(key)
        .append(":")
        .append(record.get(key).asNode().toString());
    }
  }

  // commit the transaction and for rollback we can use beginTransaction.rollback();
  tx.commit();

  // close the driver
  driver.close();
}
```

Se la modalità di accesso è impostata su `READ` nella configurazione della sessione, Neptune valuta le transazioni esplicite di sola lettura come query di sola lettura in base alla semantica di isolamento `SNAPSHOT`. Notare che le repliche di lettura accettano solo query di sola lettura.

Se non si esegue una configurazione di sessione, le transazioni esplicite di sola lettura vengono elaborate per impostazione predefinita con l'isolamento delle query di mutazione, quindi è importante eseguire una configurazione di sessione che imposti esplicitamente la modalità di accesso su `READ`.

In caso di errore, le query esplicite di sola lettura vengono ripetute per impostazione predefinita.

Query sulle transazioni di mutazione Bolt

Come per le query di sola lettura, esistono vari modi in cui è possibile elaborare le query di mutazione, con diversi modelli di transazione e livelli di isolamento, come segue:

Query sulle transazioni di mutazione implicite

Di seguito è illustrato un esempio di transazione di mutazione implicita:

```
public void executeWriteImplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";
```

```
// create node with label as label and properties.
final String WRITE_QUERY = "CREATE (n:label {name : 'foo'})";

// Read the vertex created with label as label.
final String READ_QUERY = "MATCH (n:label) RETURN n";

// create the driver
final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
  Config.builder()
        .withEncryption()
        .withTrustStrategy(TrustStrategy.trustSystemCertificates())
        .build());

// create the session config
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();

final StringBuilder resultCollector = new StringBuilder();

// run the query as access mode write
driver.session(sessionConfig).writeTransaction(new TransactionWork<String>()
{
  @Override
  public String execute(final Transaction tx)
  {
    // execute the write query and consume the result.
    tx.run(WRITE_QUERY).consume();

    // read the vertex written in the same transaction
    final List<Record> list = tx.run(READ_QUERY).list();

    // read the result
    for (final Record record : list)
    {
      for (String key : record.keys())
      {
        resultCollector
          .append(key)
          .append(":")
          .append(record.get(key).asNode().toString());
      }
```

```
      }
      return resultCollector.toString();
    }
  }); // at the end, the transaction is automatically committed.

  // close the driver.
  driver.close();
}
```

Le letture effettuate come parte delle query di mutazione vengono eseguite in base all'isolamento `READ COMMITTED` con le consuete garanzie per le transazioni di mutazione di Neptune.

Indipendentemente dal fatto che si esegua o meno specificamente una configurazione di sessione, la transazione viene sempre considerata come una transazione di scrittura.

Per i conflitti, consulta Risoluzione dei conflitti tramite timeout di attesa di blocco.

Query sulle transazioni di mutazione con commit automatico

Le query di mutazione con commit automatico ereditano lo stesso comportamento delle transazioni di mutazione implicite.

Se non si esegue una configurazione di sessione, la transazione viene considerata come una transazione di scrittura per impostazione predefinita.

In caso di errore, le query di mutazione con commit automatico non vengono ripetute automaticamente.

Query sulle transazioni di mutazione esplicite

Di seguito è illustrato un esempio di transazione di mutazione esplicita:

```
public void executeWriteExplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // create node with label as label and properties.
  final String WRITE_QUERY = "CREATE (n:label {name : 'foo'})";

  // Read the vertex created with label as label.
  final String READ_QUERY = "MATCH (n:label) RETURN n";

  // create the driver
```

```
final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
  Config.builder()
        .withEncryption()
        .withTrustStrategy(TrustStrategy.trustSystemCertificates())
        .build());

// create the session config
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();

final StringBuilder resultCollector = new StringBuilder();

final Session session = driver.session(sessionConfig);

// run the query as access mode write
final Transaction tx = driver.session(sessionConfig).beginTransaction();

// execute the write query and consume the result.
tx.run(WRITE_QUERY).consume();

// read the result from the previous write query in a same transaction.
final List<Record> list = tx.run(READ_QUERY).list();

// read the result
for (final Record record : list)
{
  for (String key : record.keys())
  {
    resultCollector
      .append(key)
      .append(":")
      .append(record.get(key).asNode().toString());
  }
}

// commit the transaction and for rollback we can use tx.rollback();
tx.commit();

// close the session
session.close();
```

```
  // close the driver.
  driver.close();
}
```

Le query di mutazione esplicita ereditano lo stesso comportamento delle transazioni di mutazione implicite.

Se non si esegue una configurazione di sessione, la transazione viene considerata come una transazione di scrittura per impostazione predefinita.

Per i conflitti, consulta Risoluzione dei conflitti tramite timeout di attesa di blocco.

Restrizioni di Neptune openCypher

Il rilascio di openCypher di Amazon Neptune non supporta ancora tutto ciò che è specificato in Cypher Query Language Reference, versione 9 come descritto in Conformità alle specifiche OpenCypher. Si prevede che i rilasci futuri risolvano molte di queste limitazioni.

Eccezioni di Neptune openCypher

Quando si usa openCypher in Amazon Neptune, possono verificarsi diverse eccezioni. Di seguito sono elencate le eccezioni più comuni che si possono ricevere dall'endpoint HTTPS o dal driver Bolt (tutte le eccezioni del driver Bolt sono segnalate come eccezioni dello stato del server):

Codice HTTP	Messaggio di errore	Recuperabile?	Soluzione	
400	(errore di sintassi, propagato direttamente dal parser openCypher)	No	Correggere la sintassi della query, quindi riprovare.	
500	`Operation terminated (out of memory)`	Sì	Rielaborare la query per aggiunger e criteri di filtro aggiuntiv	

Codice HTTP	Messaggio di errore	Recuperabile?	Soluzione	
			i per ridurre la memoria richiesta	
500	Operazione terminata (scadenza superata)	Sì	Aumentare il timeout della query nel gruppo di parametri del cluster database o ripetere la richiesta.	
500	Operazione terminata (annullata dall'utente)	Sì	Riprova la richiesta .	
500	Il ripristino del database è in corso. Riprova a eseguire la query dopo che il cluster diventa disponibile.	Sì	Riprovare quando il ripristino è stato completato.	
500	Operazione non riuscita a causa di operazioni simultanee in conflitto (riprova) . Le transazioni sono attualmente in fase di rollback.	Sì	Riprovare utilizzando una strategia di backoff esponenziale e ripetizione dei tentativi.	

Codice HTTP	Messaggio di errore	Recuperabile?	Soluzione	
400	Eccezione per funzionalità/operazione *(nome dell'operazione)* non supportata	No	L'operazione specificata non è supportata.	
400	Tentativo di aggiornamento di openCypher su una replica di sola lettura	No	Cambiare l'endpoint di destinazione con l'endpoint di scrittura.	
400	Malformed QueryException (Neptune non mostra lo stato interno del parser)	No	Correggere la sintassi della query e riprovare.	
400	Impossibile eliminare il nodo, perché ha ancora delle relazioni. Per eliminare questo nodo, devi prima eliminare le sue relazioni.	No	Invece di usare `MATCH (n) DELETE n` usare `MATCH(n) DETACH DELETE(n)`	

Codice HTTP	Messaggio di errore	Recuperabile?	Soluzione	
400	Operazione non valida: tentativo di rimuovere l'ultima etichetta di un nodo. Un nodo deve avere almeno un'etichetta.	No	Neptune richiede che tutti i nodi abbiano almeno un'etichetta. Se i nodi vengono creati senza un'etichetta esplicita, viene assegnata un'etichetta predefinita `vertex`. Modificare la logica della query e/o dell'applicazione in modo da non eliminare l'ultima etichetta. L'etichetta singleton di un nodo può essere aggiornata impostando una nuova etichetta e quindi rimuovendo quella vecchia.	

Codice HTTP	Messaggio di errore	Recuperabile?	Soluzione	
500	Il numero massimo di richieste è stato violato, Configure dQueueCapacity = {} per ConnId = {}	Sì	Attualmente è possibile elaborare solo 8.192 richieste simultane e, indipende ntemente dallo stack e dal protocollo.	
500	Limite massimo di connessioni violato.	Sì	Sono consentit e solo 1.000 connessioni Bolt simultanee per istanza (per HTTP non esiste alcun limite).	
400	Era previsto uno dei seguenti elementi [nodo, relazione o percorso] ed è stato ottenuto un valore letterale	No	Verificare che siano stati passati gli argomenti corretti, correggere la sintassi della query e riprovare .	

Codice HTTP	Messaggio di errore	Recuperabile?	Soluzione	
400	Il valore della proprietà deve essere un valore letterale semplice. Oppure: era prevista una mappa per le proprietà Set ma non ne è stata trovata una.	No	Una clausola SET accetta solo valori letterali semplici, non tipi compositi.	
400	L'entità passata per l'elimina zione non è stata trovata	No	Verificare che l'entità che si sta cercando di eliminare esista nel database.	
400	L'utente non ha accesso al database.	No	Controllare la policy sul ruolo IAM utilizzato.	
400	Non è stato passato alcun token come parte della richiesta	No	Un token firmato correttamente deve essere passato come parte della richiesta di query su un cluster abilitato per IAM.	

Codice HTTP	Messaggio di errore	Recuperabile?	Soluzione	
400	Il messaggio di errore viene propagato.	No	Contatta l' AWS assistenza con l'ID della richiesta.	
500	Operazione terminata (errore interno)	Sì	Contatta l' AWS assistenza con l'ID della richiesta.	

Accesso al grafo Neptune con SPARQL

SPARQL è un linguaggio di query per l'RDF (Resource Description Framework), un formato di dati a grafo progettato per il Web. Amazon Neptune è compatibile con SPARQL 1.1. Questo ti consente di connetterti a un'istanza database Neptune ed eseguire query sul grafo utilizzando il linguaggio di query descritto nella specifica SPARQL 1.1 Query Language.

Una query in SPARQL consiste di una clausola `SELECT` per specificare le variabili da restituire e una clausola `WHERE` per specificare i dati da abbinare nel grafo. Per ulteriori informazioni sulle query SPARQL, vedere Writing Simple Queries (Scrittura di query semplici) in SPARQL 1.1 Query Language.

> **Important**
>
> Per caricare i dati, `SPARQL UPDATE INSERT` potrebbe funzionare bene per un set di dati di piccole dimensioni, ma se è necessario caricare una notevole quantità di dati da un file, vedere Uso dello strumento di caricamento in blocco Amazon Neptune per importare i dati.

Per ulteriori informazioni sulle specifiche dell'implementazione SPARQL di Neptune, consulta Conformità agli standard SPARQL.

Prima di iniziare, devi disporre di quanto segue:

- Istanza database Neptune. Per informazioni sulla creazione di un'istanza database Neptune, consulta Creazione di un nuovo cluster database Neptune.
- Istanza Amazon EC2 nello stesso cloud privato virtuale (VPC) dell'istanza database Neptune.

Argomenti

Utilizzo della console RDF4J per la connessione a un'istanza database Neptune

La console RDF4J consente di sperimentare con i grafici e le query del Resource Description Framework (RDF) in un ambiente REPL (loop). read-eval-print

Puoi aggiungere il database a grafo remoto come archivio ed interrogarlo dalla console RDF4J. In questa sezione è illustrata la configurazione della console RDF4J per la connessione remota a un'istanza database Neptune.

Per connettersi a Neptune utilizzando la console RDF4J

1. Scaricare il kit SDK di RDF4J dalla pagina Download del sito Web di RDF4J.
2. Decomprimere il file zip SDK di RDF4J.
3. In un terminale, andare alla directory SDK di RDF4J e immettere il comando seguente per eseguire la console RDF4J:

```
bin/console.sh
```

Verrà visualizzato un output simile al seguente:

```
14:11:51.126 [main] DEBUG o.e.r.c.platform.PlatformFactory - os.name = linux
14:11:51.130 [main] DEBUG o.e.r.c.platform.PlatformFactory - Detected Posix
 platform
Connected to default data directory
RDF4J Console 3.6.1

3.6.1
Type 'help' for help.
>
```

Ora ti trovi al prompt >. Questo è il prompt generale per la console RDF4J. Puoi usare questo prompt per la configurazione degli archivi e altre operazioni. Un archivio ha il proprio prompt per l'esecuzione di query.

4. Al prompt >, digita quanto segue per creare un repository SPARQL per l'istanza database Neptune:

```
create sparql
```

5. La console RDF4J ti richiederà i valori per le variabili necessarie a connettersi all'endpoint di SPARQL.

```
Please specify values for the following variables:
```

Specifica i seguenti valori:

Nome variabile	Valore
SPARQL query endpoint	*https your-neptune-endpoint :// : porta /sparql*
SPARQL update endpoint	*https://your-neptune-endpoint : porta /sparql*

Local repository ID [endpoint@localhost]	`neptune`
Repository title [SPARQL endpoint repository @localhost]	`Istanza database Neptune`

Per informazioni su come trovare l'indirizzo dell'istanza database Neptune, consulta la sezione Connessione agli endpoint Amazon Neptune.

Se l'operazione ha esito positivo, visualizzerai il messaggio seguente:

```
Repository created
```

6. Al prompt >, immetti quanto segue per connetterti all'istanza database Neptune:

```
open neptune
```

Se l'operazione ha esito positivo, visualizzerai il messaggio seguente:

```
Opened repository 'neptune'
```

Ora ti trovi al prompt `neptune>`. Da questo prompt, puoi eseguire le query sul grafo Neptune.

> **Note**
>
> Una volta aggiunto il repository, alla successiva esecuzione di `bin/console.sh` puoi eseguire immediatamente il comando `open neptune` per connetterti all'istanza database Neptune.

7. Al `neptune>` prompt, immettete quanto segue per eseguire una query SPARQL che restituisca fino a 10 triple (subject-predicate-object) nel grafico utilizzando la `?s ?p ?o` query con un limite di 10. Per eseguire una query su qualcos'altro, sostituire il testo dopo il comando `sparql` con un'altra query SPARQL.

```
sparql select ?s ?p ?o where {?s ?p ?o} limit 10
```

Utilizzo di RDF4J Workbench per connettersi a un'istanza database Neptune

Questa sezione illustra la connessione a un'istanza database Amazon Neptune tramite RDF4J Workbench e RDF4J Server. RDF4J Server è obbligatorio perché agisce da proxy tra l'endpoint SPARQL HTTP REST di Neptune e RDF4J Workbench.

RDF4J Workbench fornisce una interfaccia facile da per sperimentare l'uso di un grafo, incluso il caricamento di file locali. Per informazioni, vedere la sezione Aggiungi nella documentazione di RDF4J.

Prerequisiti

Prima di iniziare, esegui queste attività:

- Installazione di Java 1.8 o versione successiva.
- Installazione di RDF4J Server e RDF4J Workbench. Per ulteriori informazioni, vedere Installazione di RDF4J Server e RDF4J Workbench.

Per connettersi a Neptune tramite RDF4J Workbench

1. In un browser Web, andare all'URL in cui la app Web RDF4J Workbench viene distribuita. Ad esempio, se utilizzi Apache Tomcat, l'URL è: https://*ec2_hostname*:8080/rdf4j-workbench/.
2. Se viene chiesto di connettersi a RDF4J Server, verificare che RDF4J Server sia installato, in esecuzione e che l'URL del server sia corretto. Quindi, passare alla fase successiva.
3. Nel riquadro a sinistra, scegliere New repository (Nuovo archivio).

   In New repository (Nuovo archivio):

   - Dall'elenco a discesa Type (Tipo), scegli SPARQL endpoint proxy (Proxy dell'endpoint SPARQL).
   - Per ID, digitare neptune.
   - Per Titolo, digita Istanza database Neptune.

   Seleziona Successivo.
4. In New repository (Nuovo archivio):

- Per SPARQL query endpoint URL (URL endpoint query SPARQL), digitare `https://your-neptune-endpoint:port/sparql`.
- Per SPARQL update endpoint URL (URL endpoint aggiornamento SPARQL), digitare `https://your-neptune-endpoint:port/sparql`.

Per informazioni su come trovare l'indirizzo dell'istanza database Neptune, consulta la sezione Connessione agli endpoint Amazon Neptune.

Scegli Crea.

5. Ora l'archivio di neptune (Neptune) viene visualizzato nell'elenco degli archivi. Potrebbero essere necessari alcuni minuti prima di poter utilizzare il nuovo archivio.
6. Nella colonna Id della tabella, scegliere il collegamento neptune.
7. Nel riquadro a sinistra, scegliere Query.

> **Note**
>
> Se le voci del menu Explore (Esplora) sono disabilitate, devi ricollegarti al Server RDF4J e selezionare nuovamente l'archivio di neptune (Neptune).
> Puoi eseguire questa operazione tramite i link [change] (modifica) nella parte superiore destra.

8. Nel campo di query, digita la query SPARQL seguente, quindi scegli Execute (Esegui).

```
select ?s ?p ?o where {?s ?p ?o} limit 10
```

L'esempio precedente restituisce fino a 10 delle triple (subject-predicate-object) del grafico utilizzando la `?s ?p ?o` query con un limite di 10.

Utilizzo di Java per connettersi a un'istanza database Neptune

Questa sezione illustra come eseguire un esempio di Java completo che si connette a un'istanza database Amazon Neptune ed esegue una query SPARQL.

Segui queste istruzioni da un'istanza Amazon EC2 nello stesso cloud privato virtuale (VPC) dell'istanza database Neptune.

Per connettersi a Neptune tramite Java

1. Installare Apache Maven sull'istanza EC2. Per aggiungere un archivio con un pacchetto Maven, immettere prima quanto segue:

   ```
   sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Per impostare il numero di versione per i pacchetti, immettere quanto segue.

   ```
   sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Potrai quindi utilizzare yum per installare Maven:

   ```
   sudo yum install -y apache-maven
   ```

2. Questo esempio è stato testato solo con Java 8. Per installare Java 8 sull'istanza EC2, digitare quanto segue:

   ```
   sudo yum install java-1.8.0-devel
   ```

3. Per impostare Java 8 come runtime predefinito sull'istanza EC2, digitare quanto segue:

   ```
   sudo /usr/sbin/alternatives --config java
   ```

   Quando richiesto, immetti il numero per Java 8.

4. Per impostare Java 8 come compilatore predefinito sull'istanza EC2, immettere quanto segue:

   ```
   sudo /usr/sbin/alternatives --config javac
   ```

   Quando richiesto, immetti il numero per Java 8.

5. In una nuova directory , creare un file `pom.xml`, quindi aprirlo in un editor di testo.

6. Copia quanto segue nel file `pom.xml` e salvalo (di solito puoi modificare i numeri di versione in base all'ultima versione stabile):

```
<project xmlns="https://maven.apache.org/POM/4.0.0" xmlns:xsi="https://
www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.amazonaws</groupId>
  <artifactId>RDFExample</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>RDFExample</name>
  <url>https://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>org.eclipse.rdf4j</groupId>
      <artifactId>rdf4j-runtime</artifactId>
      <version>3.6</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>exec-maven-plugin</artifactId>
          <version>1.2.1</version>
          <configuration>
            <mainClass>com.amazonaws.App</mainClass>
          </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

> **Note**
>
> Se stai modificando un progetto Maven esistente, la dipendenza necessaria è evidenziata nel codice precedente.

7. Per creare sottodirectory per il codice sorgente di esempio (`src/main/java/com/amazonaws/`), immettere quanto segue nella riga di comando:

```
mkdir -p src/main/java/com/amazonaws/
```

8. Nella directory `src/main/java/com/amazonaws/`, creare un file denominato `App.java`, quindi aprirlo in un editor di testo.

9. Copiare quanto segue nel file `App.java`. Sostituisci *`your-neptune-endpoint`*con l'indirizzo della tua istanza DB Neptune.

> **Note**
>
> Per informazioni su come trovare il nome host dell'istanza database Neptune, consulta la sezione Connessione agli endpoint Amazon Neptune.

```
package com.amazonaws;

import org.eclipse.rdf4j.repository.Repository;
import org.eclipse.rdf4j.repository.http.HTTPRepository;
import org.eclipse.rdf4j.repository.sparql.SPARQLRepository;

import java.util.List;
import org.eclipse.rdf4j.RDF4JException;
import org.eclipse.rdf4j.repository.RepositoryConnection;
import org.eclipse.rdf4j.query.TupleQuery;
import org.eclipse.rdf4j.query.TupleQueryResult;
import org.eclipse.rdf4j.query.BindingSet;
import org.eclipse.rdf4j.query.QueryLanguage;
import org.eclipse.rdf4j.model.Value;

public class App
{
    public static void main( String[] args )
```

```
    {
        String sparqlEndpoint = "https://your-neptune-endpoint:port/sparql";
        Repository repo = new SPARQLRepository(sparqlEndpoint);
        repo.initialize();

        try (RepositoryConnection conn = repo.getConnection()) {
           String queryString = "SELECT ?s ?p ?o WHERE { ?s ?p ?o } limit 10";

           TupleQuery tupleQuery = conn.prepareTupleQuery(QueryLanguage.SPARQL,
 queryString);

           try (TupleQueryResult result = tupleQuery.evaluate()) {
              while (result.hasNext()) {  // iterate over the result
                   BindingSet bindingSet = result.next();

                   Value s = bindingSet.getValue("s");
                   Value p = bindingSet.getValue("p");
                   Value o = bindingSet.getValue("o");

                   System.out.print(s);
                   System.out.print("\t");
                   System.out.print(p);
                   System.out.print("\t");
                   System.out.println(o);
              }
           }
        }
    }
}
```

10. Per compilare ed eseguire l'esempio, usare il comando Maven seguente:

```
mvn compile exec:java
```

L'esempio precedente restituisce fino a 10 delle triple (subject-predicate-object) nel grafico utilizzando la `?s ?p ?o` query con un limite di 10. Per eseguire una query su qualcos'altro, sostituire la query con un'altra query SPARQL .

L'iterazione dei risultati dell'esempio stampa il valore di ogni variabile restituita. L'oggetto `Value` viene convertito in `String` e quindi stampato. Se modifichi la parte `SELECT` della query, dovrai modificare il codice.

API SPARQL HTTP

Le richieste SPARQL HTTP vengono accettate negli endpoint seguenti: `https://your-neptune-endpoint:port/sparql`

Per ulteriori informazioni sulla connessione ad Amazon Neptune con SPARQL, vedi Accesso al grafo Neptune con SPARQL.

Per ulteriori informazioni sui protocolli SPARQL e il linguaggio di query, vedi le specifiche SPARQL 1.1 Protocol e SPARQL 1.1 Query Language.

I seguenti argomenti forniscono informazioni sui formati di serializzazione SPARQL RDF e su come utilizzare l'API HTTP di SPARQL con Neptune.

Indice

Utilizzo dell'endpoint HTTP REST per connettersi a un'istanza database Neptune

Amazon Neptune fornisce un endpoint HTTP per le query SPARQL. L'interfaccia REST è compatibile con la versione 1.1 di SPARQL.

> **Important**
>
> Rilascio: 1.0.4.0 (12/10/2020) ha reso TLS 1.2 e HTTPS obbligatori per tutte le connessioni ad Amazon Neptune. Non è più possibile connettersi a Neptune utilizzando HTTP non protetto o utilizzando HTTPS con una versione di TLS precedente alla 1.2.

Le istruzioni seguenti illustrano come connettersi a un endpoint di SPARQL utilizzando il comando curl, collegandosi tramite HTTPS e usando la sintassi HTTP. Segui queste istruzioni da un'istanza Amazon EC2 nello stesso cloud privato virtuale (VPC) dell'istanza database Neptune.

L'endpoint HTTP per le query SPARQL in un'istanza database Neptune è: `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.

> **Note**
>
> Per informazioni su come trovare il nome host dell'istanza database Neptune, consulta la sezione Connessione agli endpoint Amazon Neptune.

QUERY tramite HTTP POST

L'esempio seguente utilizza curl per inviare una SPARQL **`QUERY`** tramite HTTP POST.

```
curl -X POST --data-binary 'query=select ?s ?p ?o where {?s ?p ?o} limit 10'
 https://your-neptune-endpoint:port/sparql
```

L'esempio precedente restituisce fino a 10 delle triple (subject-predicate-object) del grafico utilizzando la `?s ?p ?o` query con un limite di 10. Per eseguire una query su qualcos'altro, sostituirla con un'altra query SPARQL .

> **Note**
>
> Il tipo di supporto MIME predefinito di una risposta è `application/sparql-results+json` per le query `SELECT` e `ASK`.
> Il tipo MIME predefinito di una risposta è `application/n-quads` per le query `CONSTRUCT` e `DESCRIBE`.
> Per un elenco dei tipi di supporto utilizzati da Neptune per la serializzazione, consulta Formati di serializzazione RDF usati da Neptune SPARQL.

UPDATE utilizzando HTTP POST

L'esempio seguente utilizza curl per inviare una SPARQL **`UPDATE`** tramite HTTP POST.

```
curl -X POST --data-binary 'update=INSERT DATA { <https://test.com/s> <https://
test.com/p> <https://test.com/o> . }' https://your-neptune-endpoint:port/sparql
```

L'esempio precedente inserisce la seguente tripla nel grafo SPARQL predefinito: `<https://test.com/s> <https://test.com/p> <https://test.com/o>`

Intestazioni HTTP finali opzionali per risposte SPARQL in più parti

> **Note**
>
> Questa funzionalità è disponibile a partire dal rilascio 1.0.3.0 del motore Neptune.

La risposta HTTP alle query e agli aggiornamenti SPARQL viene spesso restituita in più parti o blocchi. Può essere difficile diagnosticare un errore che si verifica dopo che una query o un aggiornamento inizia a inviare questi blocchi, soprattutto perché il primo arriva con un codice di stato HTTP di `200`.

A meno che non si richiedano esplicitamente le intestazioni finali, Neptune segnala tale errore solo aggiungendo un messaggio di errore al corpo del messaggio, che di solito è danneggiato.

Per facilitare il rilevamento e la diagnosi di questo tipo di problema, puoi includere nella richiesta un'intestazione transfer-encoding (TE) trailers (`te: trailers`) (vedi, ad esempio, la pagina MDN sulle intestazioni di richiesta TE). In questo modo Neptune includerà due nuovi campi di intestazione nelle intestazioni finali dei blocchi di risposta:

- `X-Neptune-Status`: contiene il codice di risposta seguito da un nome breve. Ad esempio, in caso di esito positivo, l'intestazione finale sarà: `X-Neptune-Status: 200 OK`. In caso di errore, il codice di risposta sarà un codice di errore del motore Neptune come `X-Neptune-Status: 500 TimeLimitExceededException`.
- `X-Neptune-Detail`: è vuoto per le richieste riuscite. In caso di errori, contiene il messaggio di errore JSON. Poiché nei valori di intestazione HTTP sono consentiti solo caratteri ASCII, la stringa JSON è codificata come URL. Inoltre, al corpo del messaggio di risposta viene anche aggiunto il messaggio di errore.

Tipi di supporti RDF usati da SPARQL in Neptune

I dati RDF (Resource Description Framework) possono essere serializzati in molti modi diversi e SPARQL può utilizzarne o produrne la maggior parte:

Formati di serializzazione RDF usati da Neptune SPARQL

- RDF/XML: serializzazione XML di RDF, definita in RDF 1.1 XML Syntax. Tipo di supporto: `application/rdf+xml`. Estensione tipica del file: `.rdf`.
- N-Triples: un formato di testo normale, basato su riga, per la codifica di un grafo RDF, definito in RDF 1.1 N-Triples. Tipo di supporto: `application/n-triples`, `text/turtle` o `text/plain`. Estensione tipica del file: `.nt`.
- N-Quads: un formato di testo normale, basato su riga, per la codifica di un grafo RDF, definito in RDF 1.1 N-Quads. Si tratta di un'estensione di N-Triples. Tipo di supporto: `application/n-quads` oppure `text/x-nquads` quando codificato con US-ASCII a 7 bit. Estensione tipica del file: `.nq`.
- Turtle: una sintassi testuale per RDF definita in RDF 1.1 Turtle che permette a un grafo RDF di essere completamente scritto in una forma di testo naturale e compatta, con abbreviazioni per modelli di utilizzo e tipi di dati comuni. Turtle offre livelli di compatibilità con il formato N-Triples nonché con la sintassi di modello triplice di SPARQL. Tipo di supporto: `text/turtle`Estensione tipica del file `.ttl`.
- TriG: una sintassi testuale per RDF definita in RDF 1.1 TriG che consente a un grafo RDF di essere completamente scritto in una forma di testo naturale e compatta, con abbreviazioni per modelli di utilizzo e tipi di dati comuni. TriG è un'estensione del formato Turtle. Tipo di supporto: `application/trig`. Estensione tipica del file: `.trig`.
- N3 (Notation3): un linguaggio di asserzione e logica definito in Notation3 (N3): A readable RDF syntax. N3 estende il modello di dati RDF aggiungendo formule (valori letterali che sono essi stessi grafi), variabili, implicazioni logiche e predicati funzionali e fornisce una sintassi testuale alternativa a RDF/XML. Tipo di supporto: `text/n3`. Estensione tipica del file: `.n3`.
- JSON-LD: un formato di serializzazione dei dati e dei messaggi definito in JSON-LD 1.0. Tipo di supporto: `application/ld+json`. Estensione tipica del file: `.jsonld`.
- TriX: una serializzazione di RDF in XML, definita in TriX: triple RDF in XML. Tipo di supporto: `application/trix`. Estensione tipica del file: `.trix`.
- SPARQL JSON Results: una serializzazione di RDF con SPARQL 1.1 Query Results JSON Format. Tipo di supporto: `application/sparql-results+json`. Estensione tipica del file: `.srj`.

- RDF4J Binary Format: un formato binario per la codifica di dati RDF, documentato in RDF4J Binary RDF Format. Tipo di supporto: `application/x-binary-rdf`.

Formati di serializzazione dei risultati SPARQL utilizzati da Neptune SPARQL

- SPARQL XML Results: un formato XML per i formati di risultati booleani e di vincolo variabili forniti dal linguaggio di query SPARQL, definito in SPARQL Query Results XML Format (Second Edition). Tipo di supporto: `application/sparql-results+xml`. Estensione tipica del file: `.srx`.
- SPARQL CSV and TSV Results: l'uso di valori separati da virgole e valori separati da tabulazione per esprimere i risultati di query SPARQL da query `SELECT`, definito in SPARQL 1.1 Query Results CSV and TSV Formats. Tipo di supporto: `text/csv` per valori separati da virgola e `text/tab-separated-values` per valori separati da schede. Estensioni tipiche del file: `.csv` per valori separati da virgola e `.tsv` per valori separati da schede.
- Binary Results Table: un formato binario per la codifica dell'output di query SPARQL. Tipo di supporto: `application/x-binary-rdf-results-table`.
- SPARQL JSON Results: una serializzazione di RDF con SPARQL 1.1 Query Results JSON Format. Tipo di supporto: `application/sparql-results+json`.

Tipi di supporto che Neptune può utilizzare per importare dati RDF

Tipi di supporto che lo strumento di caricamento in blocco Neptune può supportare

- N-Triples
- N-QUAD
- RDF/XML
- Turtle

Tipi di supporto che possono essere importati da SPARQL UPDATE LOAD

- N-Triples
- N-QUAD
- RDF/XML
- Turtle
- TriG
- N3

- JSON-LD

Tipi di supporto utilizzabili da Neptune per esportare risultati di query

Per specificare il formato di output per una risposta a una query di SPARQL, invia un'intestazione `"Accept: media-type"` con la richiesta di query. Per esempio:

```
curl -H "Accept: application/nquads" ...
```

Tipi di supporto RDF che SPARQL SELECT può produrre da Neptune

- SPARQL JSON Results (impostazione predefinita)
- SPARQL XML Results
- Binary Results Table (tipo di supporto: `application/x-binary-rdf-results-table`)
- Comma-Separated Values (CSV)
- Tab-Separated Values (TSV)

Tipi di supporto RDF che SPARQL ASK può produrre da Neptune

- SPARQL JSON Results (impostazione predefinita)
- SPARQL XML Results
- Boolean (tipo di supporto: `text/boolean`, ovvero "true" o "false")

Tipi di supporto RDF che SPARQL CONSTRUCT può produrre da Neptune

- N-QUAD (impostazione predefinita)
- RDF/XML
- JSON-LD
- N-Triples
- Turtle
- N3
- TriX
- TriG
- SPARQL JSON Results

- RDF4J Binary RDF Format

Tipi di supporto RDF che SPARQL DESCRIBE può produrre da Neptune

- N-QUAD (impostazione predefinita)
- RDF/XML
- JSON-LD
- N-Triples
- Turtle
- N3
- TriX
- TriG
- SPARQL JSON Results
- RDF4J Binary RDF Format

Utilizzo di SPARQL UPDATE LOAD per l'importazione di dati in Neptune

La sintassi del comando SPARQL UPDATE LOAD è specificata nella raccomandazione SPARQL 1.1 Update:

```
LOAD SILENT (URL of data to be loaded) INTO GRAPH (named graph into which to load the
 data)
```

- **`SILENT`**: (facoltativo) fa sì che l'operazione restituisca un esito positivo anche se si è verificato un errore durante l'elaborazione.

  Ciò può essere utile quando una singola transazione contiene più istruzioni come `"LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;"` e si desidera che la transazione venga completata anche se non è stato possibile elaborare alcuni dati remoti.
- *`URL of data to be loaded`*: (obbligatorio) specifica un file di dati remoti contenente i dati da caricare in un grafo.

  Il file remoto deve avere una delle seguenti estensioni:

  - `.nt` per NTriples.
  - `.nq` per NQuads.

- `.trig` per Trig.
- `.rdf` per RDF/XML.
- `.ttl` per Turtle.
- `.n3` per N3.
- `.jsonld` per JSON-LD.

- **`INTO GRAPH`***`(named graph into which to load the data)`*: (facoltativo) specifica il grafo in cui devono essere caricati i dati.

  Neptune associa ogni tripla a un grafo nominato. È possibile specificare il grafo nominato predefinito utilizzando l'URI di fallback del grafo nominato, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`, in questo modo:

  ```
  INTO GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
  ```

> **Note**
>
> Quando devi caricare molti dati, è consigliabile utilizzare lo strumento di caricamento in blocco Neptune anziché UPDATE LOAD. Per ulteriori informazioni sullo strumento di caricamento in blocco, consulta Uso dello strumento di caricamento in blocco Amazon Neptune per importare i dati.

Puoi utilizzare `SPARQL UPDATE LOAD` per caricare dati direttamente da Amazon S3 o da file ottenuti da un server Web in self-hosting. Le risorse da caricare devono trovarsi nella stessa regione del server Neptune e l'endpoint per le risorse deve essere consentito nel VCP. Per informazioni su come creare un endpoint Amazon S3, vedi Creazione di un endpoint VPC Amazon S3.

Tutti gli URI `SPARQL UPDATE LOAD` devono iniziare con `https://`. Sono inclusi gli URL Amazon S3.

Diversamente dallo strumento di caricamento in blocco Neptune, una chiamata a `SPARQL UPDATE LOAD` è completamente transazionale.

Caricare file da Amazon S3 direttamente in Neptune tramite SPARQL UPDATE LOAD

Poiché Neptune non consente di passare un ruolo IAM ad Amazon S3 quando si utilizza SPARQL UPDATE LOAD, il bucket Amazon S3 in questione deve essere pubblico oppure è necessario utilizzare un URL Amazon S3 prefirmato nella query LOAD.

Per generare un URL prefirmato per un file Amazon S3, puoi usare AWS CLI un comando come questo:

```
aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
 data to load)
```

Quindi puoi utilizzare l'URL prefirmato risultante nel comando `LOAD`:

```
curl https://(a Neptune endpoint URL):8182/sparql \
  --data-urlencode 'update=load (pre-signed URL of the remote Amazon S3 file of data to
 be loaded) \
                           into graph (named graph)'
```

Per ulteriori informazioni, consulta Autenticazione delle richieste: utilizzo dei parametri di query. La documentazione di Boto3 mostra come usare uno script Python per generare un URL prefirmato.

Inoltre, il tipo di contenuto dei file da caricare deve essere impostato correttamente.

1. Imposta il tipo di contenuto dei file quando li carichi in Amazon S3 usando il parametro `-metadata`, ad esempio:

   ```
   aws s3 cp test.nt s3://bucket-name/my-plain-text-input/test.nt --metadata Content-
   Type=text/plain
   aws s3 cp test.rdf s3://bucket-name/my-rdf-input/test.rdf --metadata Content-
   Type=application/rdf+xml
   ```

2. Conferma che sia presente l'informazione sul tipo di supporto. Esegui:

   ```
   curl -v bucket-name/folder-name
   ```

   L'output di questo comando mostra le informazioni sul tipo di supporto impostate durante il caricamento dei file.

3. Quindi, puoi utilizzare il comando `SPARQL UPDATE LOAD` per importare questi file in Neptune:

```
curl https://your-neptune-endpoint:port/sparql \
  -d "update=LOAD <https://s3.amazonaws.com/bucket-name/my-rdf-input/test.rdf>"
```

I passaggi precedenti funzionano solo per un bucket Amazon S3 pubblico o per un bucket a cui si accede utilizzando un URL Amazon S3 prefirmato nella query LOAD.

È inoltre possibile impostare un server proxy Web per il caricamento da un bucket Amazon S3 privato, come illustrato di seguito:

Utilizzare un server Web per caricare file in Neptune con SPARQL UPDATE LOAD

1. Installa un server Web su una macchina in esecuzione nel VPC che ospita Neptune e i file da caricare. Ad esempio, usando Amazon Linux, puoi installare Apache come segue:

   ```
   sudo yum install httpd mod_ssl
   sudo /usr/sbin/apachectl start
   ```

2. Definisci il tipo/i MIME dei contenuti dei file RDF che andrai a caricare. SPARQL utilizza l'intestazione `Content-type` inviata dal server Web per determinare il formato di input dei contenuti, pertanto è necessario definire i tipi MIME per il server Web.

   Ad esempio, supponiamo che desideri utilizzare le seguenti estensioni di file per identificare i formati di file:

   - `.nt` per NTriples.
   - `.nq` per NQuads.
   - `.trig` per Trig.
   - `.rdf` per RDF/XML.
   - `.ttl` per Turtle.
   - `.n3` per N3.
   - `.jsonld` per JSON-LD.

   Se stai usando Apache 2 come server Web, dovrai modificare il file `/etc/mime.types` e aggiungere i seguenti tipi:

   ```
   text/plain nt
   ```

```
application/n-quads nq
application/trig trig
application/rdf+xml rdf
application/x-turtle ttl
text/rdf+n3 n3
application/ld+json jsonld
```

3. Conferma che la mappatura del tipo MIME funziona. Una volta che il server Web è in esecuzione e ospita i file RDF nel formato di tua scelta, puoi testare la configurazione inviando una richiesta al server Web dal tuo host locale.

   Ad esempio, potresti inviare una richiesta di questo tipo:

   ```
   curl -v http://localhost:80/test.rdf
   ```

   Quindi, nell'output dettagliato di `curl`, dovresti visualizzare una riga come questa:

   ```
   Content-Type: application/rdf+xml
   ```

   Questo dimostra che la mappatura del tipo di contenuto è stata definita correttamente.

4. Ora puoi caricare i dati usando il comando SPARQL UPDATE:

   ```
   curl https://your-neptune-endpoint:port/sparql \
       -d "update=LOAD <http://web_server_private_ip:80/test.rdf>"
   ```

> **Note**
>
> Utilizzando `SPARQL UPDATE LOAD` può attivare un timeout sul server Web quando il file di origine che viene caricato è grande. Neptune elabora i dati del file mentre vengono trasmessi in streaming e per un file di grandi dimensioni che può richiedere più tempo del timeout configurato sul server. Questo a sua volta potrebbe causare la chiusura della connessione del server, che può causare il seguente messaggio di errore quando Neptune incontra un EOF imprevisto nel flusso:
>
> ```
> {
>   "detailedMessage":"Invalid syntax in the specified file",
>   "code":"InvalidParameterException"
> }
> ```

Se si riceve questo messaggio e non si ritiene che il file di origine contenga una sintassi non valida, provare ad aumentare le impostazioni di timeout sul server Web. È inoltre possibile diagnosticare il problema abilitando i registri di debug sul server e cercando i timeout.

Utilizzo di SPARQL UPDATE UNLOAD per eliminare i dati da Neptune

Neptune fornisce anche un'operazione SPARQL personalizzata, `UNLOAD`, per rimuovere i dati specificati in un'origine remota. `UNLOAD` può essere considerata una controparte dell'operazione `LOAD`. La sintassi è:

> **Note**
>
> Questa funzionalità è disponibile a partire dal rilascio 1.0.4.1 del motore Neptune.

```
UNLOAD SILENT (URL of the remote data to be unloaded) FROM GRAPH (named graph from
 which to remove the data)
```

- **`SILENT`**: (facoltativo) fa sì che l'operazione restituisca un esito positivo anche se si è verificato un errore durante l'elaborazione dei dati.

  Ciò può essere utile quando una singola transazione contiene più istruzioni come "`LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;`" e si desidera che la transazione venga completata anche se non è stato possibile elaborare alcuni dati remoti.
- *`URL of the remote data to be unloaded`*: (obbligatorio) specifica un file di dati remoti contenente i dati da scaricare da un grafo.

  Il file remoto deve avere una delle seguenti estensioni (sono gli stessi formati supportati da UPDATE-LOAD):
  - `.nt` per NTriples.
  - `.nq` per NQuads.
  - `.trig` per Trig.
  - `.rdf` per RDF/XML.
  - `.ttl` per Turtle.
  - `.n3` per N3.

- `.jsonld` per JSON-LD.

  Tutti i dati contenuti in questo file verranno rimossi dal cluster database dall'operazione `UNLOAD`.

  Qualsiasi autenticazione Amazon S3 deve essere inclusa nell'URL affinché i dati vengano scaricati. Puoi prefirmare un file Amazon S3 e quindi utilizzare l'URL risultante per accedervi in modo sicuro. Per esempio:

```
aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
 data to unload)
```

  Quindi:

```
curl https://(a Neptune endpoint URL):8182/sparql \
  --data-urlencode 'update=unload (pre-signed URL of the remote Amazon S3 data to be
 unloaded) \
                           from graph (named graph)'
```

  Per ulteriori informazioni, consulta Autenticazione delle richieste: utilizzo dei parametri di query.

- **`FROM GRAPH`** *`(named graph from which to remove the data)`*: (facoltativo) specifica il grafo nominato da cui devono essere scaricati i dati remoti.

  Neptune associa ogni tripla a un grafo nominato. È possibile specificare il grafo nominato predefinito utilizzando l'URI di fallback del grafo nominato, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`, in questo modo:

```
FROM GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
```

Nello stesso modo in cui `LOAD` corrisponde a `INSERT DATA { (inline data) }`, `UNLOAD` corrisponde a `DELETE DATA { (inline data) }`. Analogamente a `DELETE DATA`, `UNLOAD` non funziona su dati che contengono nodi vuoti.

Ad esempio, se un server Web locale serve un file denominato `data.nt` che contiene le seguenti 2 triple:

```
<http://example.org/resource#a> <http://example.org/resource#p> <http://example.org/
resource#b> .
```

```
<http://example.org/resource#a> <http://example.org/resource#p> <http://example.org/
resource#c> .
```

Il comando UNLOAD seguente eliminerà queste due triple dal grafo nominato <http://example.org/graph1>:

```
UNLOAD <http://localhost:80/data.nt> FROM GRAPH <http://example.org/graph1>
```

Ciò avrà lo stesso effetto dell'utilizzo del comando DELETE DATA seguente:

```
DELETE DATA {
  GRAPH <http://example.org/graph1> {
    <http://example.org/resource#a> <http://example.org/resource#p> <http://
example.org/resource#b> .
    <http://example.org/resource#a> <http://example.org/resource#p> <http://
example.org/resource#c> .
  }
}
```

Eccezioni generate dal comando **UNLOAD**

- **InvalidParameterException**: sono presenti nodi vuoti nei dati. Stato HTTP: 400 Richiesta non valida.

  Messaggio: `Blank nodes are not allowed for UNLOAD`

- **InvalidParameterException**: sintassi non corretta nei dati. Stato HTTP: 400 Richiesta non valida.

  Messaggio: `Invalid syntax in the specified file.`

- **UnloadUrlAccessDeniedException** : accesso negato. Stato HTTP: 400 Richiesta non valida.

  Messaggio: `Update failure: Endpoint` *`(Neptune endpoint)`* `reported access denied error. Please verify access.`

- **`BadRequestException`** : non è possibile recuperare i dati remoti. Stato HTTP: 400 Richiesta non valida.

  Messaggio: (dipende dalla risposta HTTP).

Hint di query SPARQL

È possibile utilizzare gli hint di query per specificare le strategie di ottimizzazione e valutazione per una determinata query SPARQL in Amazon Neptune.

Gli hint di query vengono espressi utilizzando i modelli di tripla aggiuntivi incorporati nella query SPARQL con le seguenti parti:

```
scope hint value
```

- ambito: determina la parte della query alla quale si applica l'hint di query, come un determinato gruppo nella query o l'intera query.
- hint: identifica il tipo di hint da applicare.
- value: determina il comportamento dell'aspetto del sistema in esame.

Gli ambiti e gli hint di query vengono esposti come termini predefiniti nello spazio dei nomi di Amazon Neptune `http://aws.amazon.com/neptune/vocab/v01/QueryHints#`. Gli esempi di questa sezione includono lo spazio dei nomi come un prefisso `hint` definito e incluso nella query:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

Ad esempio, la seguente procedura mostra come includere un hint `joinOrder` in una query `SELECT`:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ... {
 hint:Query hint:joinOrder "Ordered" .
 ...
}
```

La query precedente indica al motore Neptune di valutare i join nella query nell'ordine specificato e disabilita qualsiasi riordinamento automatico.

Considera le informazioni seguenti durante l'utilizzo dei suggerimenti di query:

- È possibile combinare diversi suggerimenti di query in una singola richiesta. Ad esempio, è possibile utilizzare l'hint di query `bottomUp` per annotare una sottoquery per una valutazione dal basso verso l'alto e un hint di query `joinOrder` per correggere il join all'interno della sottoquery.
- È possibile utilizzare lo stesso hint di query più volte, in diversi ambiti non sovrapposti.
- I suggerimenti di query sono suggerimenti. Sebbene il motore di query generalmente miri a considerare determinati suggerimenti di query, potrebbe anche ignorarli.
- Per i suggerimenti di query si applica la conservazione della semantica. L'aggiunta di un hint di query non modifica l'output della query (ad eccezione dell'ordine dei potenziali risultati quando non vengono fornite garanzie di ordinamento, ovvero quando l'ordine dei risultati non è esplicitamente applicato utilizzando ORDER BY).

Le seguenti sezioni forniscono ulteriori informazioni sugli hint di query disponibili e il relativo utilizzo in Neptune.

Argomenti

Ambito degli hint di query SPARQL in Neptune

La tabella che segue mostra gli ambiti disponibili, gli hint associati e le descrizioni per gli hint di query SPARQL in Amazon Neptune. Il prefisso `hint` in queste voci rappresenta lo spazio dei nomi Neptune per gli hint:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

Ambito	Hint supportato	Descrizione
`hint:Query`	joinOrder	L'hint di query si applica all'intera query.
`hint:Query`	queryTimeout	Il valore di timeout si applica all'intera query.
`hint:Query`	rangeSafe	La promozione dei tipi è disabilitata per l'intera query.
`hint:Query`	queryId	Il valore dell'ID della query si applica all'intera query.
`hint:Query`	useDFE	L'uso del motore DFE è abilitato (o disabilitato) per l'intera query.
`hint:Group`	joinOrder	L'hint di query si applica agli elementi di primo livello nel gruppo specificato, ma non agli elementi nidificati (come le sottoquery) o agli elementi padre.
`hint:SubQuery`	evaluationStrategy	l'hint viene specificato e applicato a una sottoquer y SELECT nidificata. La sottoquery viene valutata in modo indipendente, senza considerare le soluzioni calcolate prima della sottoquer y.

Hint di query SPARQL `joinOrder`

Quando invii una query SPARQL, il motore di query Amazon Neptune controlla la struttura della query. Riordina le parti della query e tenta di ridurre al minimo la quantità di lavoro richiesta per la valutazione e il tempo di risposta della query.

Ad esempio, una sequenza di modelli di tripla collegati non viene generalmente valutata nell'ordine specificato. Viene riordinato utilizzando l'euristica e le statistiche come la selettività dei singoli modelli e il modo in cui sono collegati tramite variabili condivise. Inoltre, se la query contiene modelli più complessi come sottoquery, FILTER o blocchi OPTIONAL o MINUS complessi, il motore di query Neptune li riordina dove possibile, secondo un ordine di valutazione efficiente.

Per richieste più complesse, l'ordine nel quale Neptune sceglie di valutare la query potrebbe non essere sempre ottimale. Ad esempio, è possibile che Neptune non consideri caratteristiche specifiche dei dati dell'istanza (come la possibilità di raggiungere nodi di potenza nel grafo) che emergono durante la valutazione delle query.

Se conosci le caratteristiche esatte dei dati e desideri indicare manualmente l'ordine di esecuzione della query, utilizza l'hint di query Neptune `joinOrder` per specificare che la query deve essere valutata nell'ordine specificato.

Sintassi dell'hint SPARQL `joinOrder`

l'hint di query `joinOrder` viene specificato come modello di tripla incluso in una query SPARQL.

Per chiarezza, la sintassi seguente utilizza un prefisso `hint` definito e incluso nella query per specificare lo spazio dei nomi dell'hint di query Neptune:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
scope hint:joinOrder "Ordered" .
```

Ambiti disponibili

- `hint:Query`
- `hint:Group`

Per ulteriori informazioni sugli ambiti dei suggerimenti di query, consulta Ambito degli hint di query SPARQL in Neptune.

Esempio dell'hint SPARQL **`joinOrder`**

Questa sezione mostra una query scritta con e senza l'hint di query `joinOrder` e le ottimizzazioni correlate.

Per questo esempio si presuppone che il set di dati contenga quanto segue:

- Una sola persona di nome `John` che `:likes` 1000 persone, inclusa `Jane`.
- Una sola persona di nome `Jane` che `:likes` 10 persone, incluso `John`.

Nessun hint di query

La seguente query SPARQL estrae tutte le coppie di persone di nome `John` e `Jane` che hanno specificato "like" reciprocamente da un set di dati di social networking:

```
PREFIX : <https://example.com/>
SELECT ?john ?jane {
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Il motore di query Neptune può valutare le istruzioni in un ordine diverso rispetto a quello scritto. Ad esempio, potrebbe scegliere di valutare nel seguente ordine:

1. Trova tutte le persone di nome `John`.
2. Trova tutte le persone collegate a `John` con un edge `:likes`.
3. Filtra questo set in base alle persone di nome `Jane`.
4. Filtra questo set in base alle persone collegate a `John` con un edge `:likes`.

In base al set di dati, la valutazione in questo ordine comporta l'estrazione di 1.000 entità nel secondo passaggio. Il terzo passaggio limita ulteriormente la valutazione fino al singolo nodo `Jane`. Il passaggio finale determina che anche `Jane` ha specificato `:likes` per il nodo `John`.

hint di query

Sarebbe preferibile iniziare con il nodo `Jane` perché ha solo 10 edge `:likes` in uscita. Ciò riduce la quantità di lavoro durante la valutazione della query evitando l'estrazione delle 1.000 entità durante il secondo passaggio.

L'esempio seguente usa l'hint di query joinOrder per assicurare che il nodo `Jane` e i relativi archi in uscita vengano elaborati per primi disabilitando il riordinamento automatico di tutti i join per la query:

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?john ?jane {
  hint:Query hint:joinOrder "Ordered" .
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Uno scenario applicabile del mondo reale potrebbe essere un'applicazione di social network in cui le persone nella rete sono classificate come influencer con molte connessioni o come utenti normali con poche connessioni. In tale scenario, puoi garantire che l'utente normale (`Jane`) venga elaborato prima dell'influencer (`John`) in una query come nell'esempio precedente.

hint di query e riordinamento

Puoi eseguire questo esempio per aumentare le tue conoscenze. Se sai che l'attributo `:name` è univoco per un singolo nodo, puoi velocizzare la query tramite il riordinamento e utilizzando l'hint di query `joinOrder`. In questo modo, i nodi univoci vengono estratti per primi.

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?john ?jane {
  hint:Query hint:joinOrder "Ordered" .
  ?person1 :name "Jane" .
  ?person2 :name "John" .
  ?person1 :likes ?person2 .
  ?person2 :likes ?person1 .
}
```

In questo caso, puoi limitare la query alle seguenti singole operazioni in ciascuna fase:

1. Trova il nodo della singola persona con `:name Jane`.

2. Trova il nodo della singola persona con `:name John`.
3. Controlla che il primo nodo sia collegato al secondo con un edge `:likes`.
4. Controlla che il secondo nodo sia collegato al primo con un edge `:likes`.

> **Important**
>
> Se scegli l'ordine non corretto, l'hint di query `joinOrder` può causare una notevole riduzione delle prestazioni. Ad esempio, l'esempio precedente sarebbe inefficiente se gli attributi `:name` non sono univoci. Se tutti i 100 nodi sono stati denominati `Jane` e tutti i 1.000 nodi sono stati denominati `John`, la query finirebbe per controllare 1.000 * 100 (100.000) coppie di edge `:likes`.

Hint di query SPARQL **`evaluationStrategy`**

L'hint di query `evaluationStrategy` indica al motore di query Amazon Neptune che il frammento della query annotata deve essere valutato dal basso verso l'alto come unità indipendente. Ciò significa che non vengono utilizzate le soluzioni dei precedenti passaggi di valutazione per calcolare il frammento della query. Il frammento di query viene valutato come un'unità indipendente e le soluzioni prodotte vengono unite al resto della query dopo che è stata calcolata.

L'uso dell'hint di query `evaluationStrategy` implica un piano di query di blocco (non pipeline), ovvero le soluzioni del frammento annotato con l'hint di query sono materializzate e memorizzate nella memoria principale. L'uso di questo hint di query può aumentare significativamente la quantità di memoria principale necessaria per valutare la query, soprattutto se il frammento di query annotato calcola un numero elevato di risultati.

Sintassi dell'hint SPARQL **`evaluationStrategy`**

l'hint di query `evaluationStrategy` viene specificato come modello di tripla incluso in una query SPARQL.

Per chiarezza, la sintassi seguente utilizza un prefisso `hint` definito e incluso nella query per specificare lo spazio dei nomi dell'hint di query Neptune:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

```
hint:SubQuery hint:evaluationStrategy "BottomUp" .
```

Ambiti disponibili

- `hint:SubQuery`

> **Note**
>
> Questo hint di query è supportato solo nelle sottoquery nidificate.

Per ulteriori informazioni sugli ambiti dei suggerimenti di query, consulta Ambito degli hint di query SPARQL in Neptune.

Esempio dell'hint SPARQL **`evaluationStrategy`**

Questa sezione mostra una query scritta con e senza l'hint di query `evaluationStrategy` e le ottimizzazioni correlate.

Per questo esempio si presuppone che il set di dati abbia le seguenti caratteristiche:

- Contiene 1.000 edge con etichetta `:connectedTo`.
- Ogni nodo `component` è connesso a una media di altri 100 nodi `component`.
- Il numero tipico di connessioni cicliche a quattro hop tra i nodi è di circa 100.

Nessun hint di query

La seguente query SPARQL estrae tutti i nodi `component` che sono ciclicamente collegati tra loro mediante quattro hop:

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
  ?component1 :connectedTo ?component2 .
  ?component2 :connectedTo ?component3 .
  ?component3 :connectedTo ?component4 .
  ?component4 :connectedTo ?component1 .
}
```

L'approccio del motore di query Neptune è quello di valutare questa query utilizzando i seguenti passaggi:

- Estrai tutti i 1.000 edge `connectedTo` nel grafico.
- Espandi per 100x (il numero di edge `connectedTo` in uscita dal component2).

  Risultati intermedi: 100.000 nodi.
- Espandi per 100x (il numero di edge `connectedTo` in uscita dal component3).

  Risultati intermedi: 10.000.000 nodi.
- Analizza i 10.000.000 nodi per il ciclo chiuso.

Ciò si traduce in un piano di query in streaming che ha una quantità costante di memoria principale.

hint di query e sottoquery

Potresti voler scambiare lo spazio di memoria principale per il calcolo accelerato. Riscrivendo la query utilizzando un hint di query `evaluationStrategy` è possibile forzare il motore a calcolare un join tra due sottoinsiemi più piccoli e materializzati.

```
PREFIX : <https://example.com/>
          PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
  {
    SELECT * WHERE {
      hint:SubQuery hint:evaluationStrategy "BottomUp" .
      ?component1 :connectedTo ?component2 .
      ?component2 :connectedTo ?component3 .
    }
  }
  {
    SELECT * WHERE {
      hint:SubQuery hint:evaluationStrategy "BottomUp" .
      ?component3 :connectedTo ?component4 .
      ?component4 :connectedTo ?component1 .
    }
  }
}
```

Invece di valutare i modelli di tripla in sequenza mentre vengono utilizzati in modo iterativo i risultati del modello di tripla precedente come input per i successivi modelli, l'hint `evaluationStrategy`

fa in modo che le due sottoquery vengano valutate in modo indipendente. Entrambe le sottoquery producono 100.000 nodi per i risultati intermedi che vengono quindi uniti per formare l'output finale.

In particolare, quando esegui Neptune nei tipi di istanza più grandi, l'archiviazione temporanea di questi due sottoinsiemi da 100.000 nodi nella memoria principale aumenta l'utilizzo della memoria in cambio di una significativa accelerazione della valutazione.

Hint di query SPARQL **`queryTimeout`**

l'hint di query `queryTimeout` specifica un timeout più breve rispetto al valore `neptune_query_timeout` impostato nel gruppo di parametri di database.

Se la query termina come risultato di questo hint, viene generata un'eccezione `TimeLimitExceededException`, con un messaggio `Operation terminated (deadline exceeded)`.

Sintassi dell'hint SPARQL **`queryTimeout`**

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ... WHERE {
    hint:Query hint:queryTimeout 10 .
    # OR
    hint:Query hint:queryTimeout "10" .
    # OR
    hint:Query hint:queryTimeout "10"^^xsd:integer .
 ...
}
```

Il valore di timeout viene espresso in millisecondi.

Il valore di timeout deve essere inferiore al valore `neptune_query_timeout` impostato nel gruppo di parametri di database. In caso contrario, viene generata un'eccezione `MalformedQueryException` con un messaggio `Malformed query: Query hint 'queryTimeout' must be less than neptune_query_timeout DB Parameter Group`.

l'hint di query `queryTimeout` dovrebbe essere specificato nella clausola `WHERE` della query principale oppure nella clausola `WHERE` di una delle sottoquery, come illustrato nell'esempio seguente.

Deve essere impostato una sola volta in tutte le query/sottoquery e sezioni aggiornamenti SPARQL (ad esempio INSERT e DELETE). In caso contrario, viene generata un'eccezione

`MalformedQueryException` con un messaggio `Malformed query: Query hint 'queryTimeout' must be set only once.`

Ambiti disponibili

L'hint `queryTimeout` può essere applicato sia alle query sia agli aggiornamenti SPARQL.

- In una query SPARQL, può comparire nella clausola WHERE della query principale o di una sottoquery.
- In un aggiornamento SPARQL, può essere impostato nella clausola INSERT, DELETE o WHERE. Se sono presenti più clausole di aggiornamento, può essere impostato solo in una di esse.

Per ulteriori informazioni sugli ambiti dei suggerimenti di query, consulta Ambito degli hint di query SPARQL in Neptune.

Esempio dell'hint SPARQL **`queryTimeout`**

Di seguito è illustrato un esempio di utilizzo di `hint:queryTimeout` nella clausola `WHERE` principale di una query `UPDATE`:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
INSERT {
    ?s ?p ?o
} WHERE {
    hint:Query hint:queryTimeout 100 .
    ?s ?p ?o .
}
```

Di seguito, `hint:queryTimeout` si trova nella clausola `WHERE` di una sottoquery:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
   ?s ?p ?o .
   {
      SELECT ?s WHERE {
         hint:Query hint:queryTimeout 100 .
         ?s ?p1 ?o1 .
      }
   }
}
```

Hint di query SPARQL **`rangeSafe`**

Usare questo hint di query per disattivare la promozione dei tipi per una query SPARQL.

Quando invii una query SPARQL che include un oggetto `FILTER` su un valore o intervallo numerico, il motore di query Neptune deve normalmente utilizzare la promozione dei tipi quando esegue la query. Ciò significa che deve esaminare i valori di ogni tipo che potrebbe contenere il valore in base al quale stai filtrando.

Ad esempio, se filtri per valori pari a 55, il motore deve cercare numeri interi pari a 55, numeri interi long pari a 55L, numeri a virgola mobile pari a 55.0 e così via. Ogni promozione di tipo richiede una ricerca aggiuntiva nell'archiviazione, che può far sì che una query apparentemente semplice richieda un tempo inaspettatamente lungo per essere completata.

Spesso la promozione dei tipi non è necessaria perché si sa in anticipo che è necessario trovare solo valori di un tipo specifico. In questo caso, puoi velocizzare notevolmente le query utilizzando l'hint di query `rangeSafe` per disattivare la promozione dei tipi.

Sintassi dell'hint SPARQL **`rangeSafe`**

L'hint di query `rangeSafe` accetta il valore `true` per disattivare la promozione dei tipi. Accetta anche il valore `false` (impostazione predefinita).

Esempio. L'esempio seguente mostra come disattivare la promozione dei tipi quando si filtra per un valore intero o maggiore di 1:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
   ?s ?p ?o .
   hint:Prior hint:rangeSafe 'true' .
   FILTER (?o > '1'^^<http://www.w3.org/2001/XMLSchema#int>)
```

Hint di query SPARQL **`queryId`**

Utilizza questo hint di query per assegnare il tuo valore queryId a una query SPARQL.

Esempio:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * WHERE {
  hint:Query hint:queryId "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

```
  {?s ?p ?o}}
```

Il valore assegnato deve essere univoco per tutte le query nel database Neptune.

Hint di query SPARQL **`useDFE`**

Utilizzare questo hint di query per abilitare l'uso del motore DFE per l'esecuzione della query. Per impostazione predefinita, Neptune non utilizza il motore DFE senza che questo hint di query sia impostato su `true`, poiché il parametro di istanza neptune_dfe_query_engine è impostato su `viaQueryHint`. Se imposti il parametro di istanza su `enabled`, il motore DFE viene utilizzato per tutte le query ad eccezione di quelle con l'hint di query `useDFE` impostato su `false`.

Esempio di abilitazione dell'uso del motore DFE per una query:

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>

SELECT ?john ?jane
{
  hint:Query hint:useDFE true .
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Hint di query SPARQL utilizzati con DESCRIBE

Una query SPARQL `DESCRIBE` fornisce un meccanismo flessibile per richiedere descrizioni di risorse. Tuttavia, le specifiche SPARQL non definiscono la semantica precisa di `DESCRIBE`.

A partire dal rilascio 1.2.0.2 del motore, Neptune supporta diverse modalità e algoritmi `DESCRIBE` adatti a diverse situazioni.

Questo set di dati di esempio può aiutare a illustrare le diverse modalità:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <https://example.com/> .

:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JohnDoe :firstName "John" .
```

```
:JaneDoe :knows _:b1 .
_:b1 :knows :RichardRoe .

:RichardRoe :knows :JaneDoe .
:RichardRoe :firstName "Richard" .

_:s1 rdf:type rdf:Statement .
_:s1 rdf:subject :JaneDoe .
_:s1 rdf:predicate :knows .
_:s1 rdf:object :JohnDoe .
_:s1 :knowsFrom "Berlin" .

:ref_s2 rdf:type rdf:Statement .
:ref_s2 rdf:subject :JaneDoe .
:ref_s2 rdf:predicate :knows .
:ref_s2 rdf:object :JohnDoe .
:ref_s2 :knowsSince 1988 .
```

Gli esempi seguenti presuppongono che venga richiesta una descrizione della risorsa `:JaneDoe` utilizzando una query SPARQL in questo modo:

```
DESCRIBE <https://example.com/JaneDoe>
```

Hint di query SPARQL **`describeMode`**

L'hint di query SPARQL `hint:describeMode` viene utilizzato per selezionare una delle seguenti modalità SPARQL `DESCRIBE` supportate da Neptune:

Modalità DESCRIBE **`ForwardOneStep`**

È possibile richiamare la modalità `ForwardOneStep` con l'hint di query `describeMode` in questo modo:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "ForwardOneStep"
}
```

La modalità `ForwardOneStep` restituisce solo gli attributi e i collegamenti diretti della risorsa da descrivere. Nel caso di esempio, ciò significa che restituisce le triple che hanno `:JaneDoe`, la risorsa da descrivere, come oggetto:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b301990159 .
```

Notare che la query DESCRIBE può restituire triple con nodi vuoti, ad esempio`_:b301990159`, che hanno ogni volta ID diversi rispetto al set di dati di input.

Modalità DESCRIBE **`SymmetricOneStep`**

`SymmetricOneStep` è la modalità DESCRIBE predefinita se non si fornisce un hint di query. Puoi anche richiamarlo esplicitamente con l'hint di query `describeMode` in questo modo:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SymmetricOneStep"
}
```

Nella semantica `SymmetricOneStep`, `DESCRIBE` restituisce gli attributi, i collegamenti diretti e i collegamenti inversi della risorsa da descrivere:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b318767375 .

_:b318767631 rdf:subject :JaneDoe .

:RichardRoe :knows :JaneDoe .

:ref_s2 rdf:subject :JaneDoe .
```

Modalità DESCRIBE Concise Bounded Description (**`CBD`**)

La modalità Concise Bounded Description (`CBD`) viene richiamata utilizzando l'hint di query `describeMode` in questo modo:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "CBD"
```

```
}
```

Nella semantica `CBD`, `DESCRIBE` restituisce la modalità Concise Bounded Description (come definito da W3C) della risorsa da descrivere:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b285212943 .
_:b285212943 :knows :RichardRoe .

_:b285213199 rdf:subject :JaneDoe .
_:b285213199 rdf:type rdf:Statement .
_:b285213199 rdf:predicate :knows .
_:b285213199 rdf:object :JohnDoe .
_:b285213199 :knowsFrom "Berlin" .

:ref_s2 rdf:subject :JaneDoe .
```

La Concise Bounded Description di una risorsa RDF (ovvero un nodo in un grafo RDF) è il sottografo più piccolo incentrato su quel nodo che può essere autonomo. In pratica ciò significa che se si pensa a questo grafo come a un albero, con il nodo designato come radice, non ci sono nodi vuoti (bnodi) come foglie di quell'albero. Poiché i bnodi non possono essere indirizzati esternamente o utilizzati nelle query successive, per l'esplorazione del grafo non è sufficiente trovare i singoli hop successivi dal nodo corrente. Devi anche andare abbastanza lontano per trovare qualcosa che possa essere usato nelle query successive (cioè qualcosa di diverso da un bnodo).

Calcolo della CBD

Dato un particolare nodo (il nodo iniziale o radice) nel grafo RDF di origine, la CBD di quel nodo viene calcolata come segue:

1. Includere nel sottografo tutte le istruzioni del grafo di origine in cui il soggetto dell'istruzione è il nodo iniziale.
2. Ricorsivamente, per tutte le istruzioni del sottografo che finora hanno un oggetto nodo vuoto, includere nel sottografo tutte le istruzioni del grafo di origine in cui il soggetto dell'istruzione è quel nodo vuoto e che non sono già incluse nel sottografo.
3. Ricorsivamente, per tutte le istruzioni incluse nel sottografo finora, per tutte le reificazioni di queste istruzioni del grafo di origine, includere la CBD a partire dal nodo `rdf:Statement` di ogni reificazione.

Si ottiene così un sottografo in cui i nodi oggetto sono riferimenti o valori letterali IRI oppure nodi vuoti che non fungono da soggetto di alcuna istruzione del grafo. Notare che la CBD non può essere calcolata utilizzando una singola query SPARQL SELECT o CONSTRUCT.

Modalità DESCRIBE Symmetric Concise Bounded Description (**`SCBD`**)

La modalità Symmetric Concise Bounded Description (`SCBD`) viene richiamata utilizzando l'hint di query `describeMode` in questo modo:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SCBD"
}
```

Nella semantica `SCBD`, `DESCRIBE` restituisce la Symmetric Concise Bounded Description della risorsa (come definito da W3C in Describing Linked Datasets with the VOID Vocabulary):

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b335544591 .
_:b335544591 :knows :RichardRoe .

:RichardRoe :knows :JaneDoe .

_:b335544847 rdf:subject :JaneDoe .
_:b335544847 rdf:type rdf:Statement .
_:b335544847 rdf:predicate :knows .
_:b335544847 rdf:object :JohnDoe .
_:b335544847 :knowsFrom "Berlin" .

:ref_s2 rdf:subject :JaneDoe .
```

Il vantaggio di CBD e SCBD rispetto alle modalità `ForwardOneStep` e `SymmetricOneStep` è che i nodi vuoti vengono sempre espansi per includere la relativa rappresentazione. Questo può essere un vantaggio importante perché non è possibile eseguire query su un nodo vuoto utilizzando SPARQL. Inoltre, le modalità CBD e SCBD considerano anche le reificazioni.

Notare che l'hint di query `describeMode` può anche far parte di una clausola `WHERE`:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE ?s
```

```
WHERE {
  hint:Query hint:describeMode "CBD" .
  ?s rdf:type <https://example.com/Person>
}
```

Hint di query SPARQL **`describeIterationLimit`**

L'hint di query SPARQL `hint:describeIterationLimit` fornisce un vincolo opzionale sul numero massimo di espansioni iterative da eseguire per gli algoritmi DESCRIBE iterativi come CBD e SCBD.

I limiti di DESCRIBE sono uniti da un operatore AND. Pertanto, se vengono specificati sia il limite di iterazione che il limite di istruzioni, entrambi i limiti devono essere soddisfatti prima che la query DESCRIBE venga interrotta.

Il valore predefinito per questo valore è 5. È possibile impostarlo su ZERO (0) per specificare NESSUN limite al numero di espansioni iterative.

Hint di query SPARQL **`describeStatementLimit`**

L'hint di query SPARQL `hint:describeStatementLimit` fornisce un vincolo opzionale sul numero massimo di istruzioni che possono essere presenti in una risposta alla query DESCRIBE. Viene applicato solo per algoritmi DESCRIBE iterativi come CBD e SCBD.

I limiti di DESCRIBE sono uniti da un operatore AND. Pertanto, se vengono specificati sia il limite di iterazione che il limite di istruzioni, entrambi i limiti devono essere soddisfatti prima che la query DESCRIBE venga interrotta.

Il valore predefinito per questo valore è 5000. È possibile impostarlo su ZERO (0) per specificare NESSUN limite al numero di istruzioni restituite.

Comportamento di SPARQL DESCRIBE rispetto al grafo predefinito

Il modulo di query SPARQL `DESCRIBE` consente di recuperare informazioni sulle risorse senza conoscere la struttura dei dati e senza dover comporre una query. Il modo in cui queste informazioni vengono assemblate dipende dall'implementazione SPARQL. Neptune fornisce diversi hint di query che richiamano diverse modalità e algoritmi da utilizzare per `DESCRIBE`.

Nell'implementazione di Neptune, indipendentemente dalla modalità, `DESCRIBE` utilizza solo i dati presenti nel grafo SPARQL predefinito. Ciò è coerente con il modo in cui SPARQL tratta i set di dati (vedi Specifying RDF Datasets nella specifica SPARQL).

In Neptune, il grafo predefinito contiene tutte le triple univoche nell'unione di tutti i grafi nominati del database, a meno che non vengano specificati grafi nominati particolari utilizzando le clausole `FROM` e/o `FROM NAMED`. Tutti i dati RDF in Neptune sono archiviati in un grafo nominato. Se una tripla viene inserita senza un contesto del grafo nominato, Neptune la archivia in un grafo nominato designato `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Quando uno o più grafi nominati vengono specificati utilizzando la clausola `FROM`, il grafo predefinito è l'unione di tutte le triple univoche presenti in tali grafi nominati. Se non è presente alcuna clausola `FROM` e sono presenti una o più clausole `FROM NAMED`, il grafo predefinito è vuoto.

Esempi di SPARQL **`DESCRIBE`**

Considerare i dati seguenti:

```
PREFIX ex: <https://example.com/>

GRAPH ex:g1 {
    ex:s ex:p1 "a" .
    ex:s ex:p2 "c" .
}

GRAPH ex:g2 {
    ex:s ex:p3 "b" .
    ex:s ex:p2 "c" .
}

ex:s ex:p3 "d" .
```

Per questa query:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM ex:g1
FROM NAMED ex:g2
WHERE {
  GRAPH ex:g2 { ?s ?p "b" . }
}
```

Neptune restituirà:

```
ex:s ex:p1 "a" .
```

```
ex:s ex:p2 "c" .
```

In questo caso, il modello di grafo `GRAPH ex:g2 { ?s ?p "b" }` viene valutato per primo, ottenendo le associazioni per `?s` e quindi la parte `DESCRIBE` viene valutata rispetto al grafo predefinito, che ora è solo `ex:g1`.

Tuttavia, per questa query:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM NAMED ex:g1
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

Neptune non restituirà nulla, perché quando è presente una clausola `FROM NAMED` senza una clausola `FROM`, il grafo predefinito è vuoto.

Nella seguente query, `DESCRIBE` viene utilizzato senza che sia presente una clausola `FROM` o `FROM NAMED`:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

In questa situazione, il grafo predefinito è composto da tutte le triple univoche nell'unione di tutti i grafi nominati nel database (formalmente, l'unione RDF), quindi Neptune restituirà:

```
ex:s ex:p1 "a" .
ex:s ex:p2 "c" .
ex:s ex:p3 "b" .
ex:s ex:p3 "d" .
```

API di stato delle query SPARQL

Per ottenere lo stato delle query SPARQL, utilizza l'operazione HTTP `GET` o `POST` per effettuare una richiesta all'endpoint `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql/status`.

Parametri della richiesta di stato della query SPARQL

queryId (opzionale)

L'ID di una query SPARQL in esecuzione. Viene mostrato solo lo stato della query specificata.

Sintassi della risposta di stato della query SPARQL

```
{
    "acceptedQueryCount": integer,
    "runningQueryCount": integer,
    "queries": [
      {
        "queryId":"guid",
        "queryEvalStats":
          {
            "subqueries": integer,
            "elapsed": integer,
            "cancelled": boolean
          },
        "queryString": "string"
      }
    ]
}
```

Valori della risposta di stato della query SPARQL

acceptedQueryCount

Numero di query accettate dall'ultimo riavvio del motore Neptune.

runningQueryCount

Il numero delle query SPARQL in esecuzione.

queries

Un elenco delle query SPARQL correnti.

queryId

Un ID GUID per la query. Neptune assegna automaticamente questo valore ID a ogni query oppure è possibile assegnare un ID personalizzato (consulta Inserimento di un ID personalizzato in una query Neptune Gremlin o SPARQL).

queryEvalStats

Le statistiche relative a questa query.

subqueries

Numero di sottoquery in questa query.

elapsed

Il numero di millisecondi durante i quali è stata eseguita la query (fino a questo momento).

cancelled

Il valore True indica che la query è stata annullata.

queryString

La query inviata.

Esempio di stato della query SPARQL

Di seguito è riportato un esempio del comando di stato che utilizza `curl` e l'operazione HTTP `GET`.

```
curl https://your-neptune-endpoint:port/sparql/status
```

Questo output mostra una sola query in esecuzione.

```
{
    "acceptedQueryCount":9,
    "runningQueryCount":1,
    "queries": [
        {
            "queryId":"fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
            "queryEvalStats":
                {
                    "subqueries": 0,
                    "elapsed": 29256,
                    "cancelled": false
                },
            "queryString": "SELECT ?s ?p ?o WHERE {?s ?p ?o}"
        }
    ]
}
```

Annullamento della query SPARQL

Per ottenere lo stato delle query SPARQL, utilizza l'operazione HTTP `GET` o `POST` per effettuare una richiesta all'endpoint `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql/status`.

Parametri della richiesta di annullamento delle query SPARQL

cancelQuery

(Obbligatorio) Indica al comando di stato di annullare una query. Questo parametro non accetta un valore.

queryId

(Obbligatorio) L'ID della query SPARQL in esecuzione da annullare.

silent

(Facoltativo) Se `silent=true`, la query in esecuzione viene annullata e il codice di risposta HTTP è 200. Se `silent` non è presente o `silent=false`, la query viene annullata con un codice di stato HTTP 500.

Esempi di annullamento di query SPARQL

Esempio 1: annullamento con **`silent=false`**

Di seguito è riportato un esempio del comando di stato che utilizza `curl` per annullare una query con il parametro `silent` impostato su `false`:

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
  -d "silent=false"
```

A meno che la query non abbia già avviato i risultati in streaming, la query annullata restituirebbe un codice HTTP 500 con una risposta come questa:

```
{
  "code": "CancelledByUserException",
  "requestId": "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47",
  "detailedMessage": "Operation terminated (cancelled by user)"
}
```

Se la query ha già restituito un codice HTTP 200 (OK) e ha iniziato lo streaming dei risultati prima di essere annullata, le informazioni sull'eccezione di timeout vengono inviate al flusso di output normale.

Esempio 2: annullamento con **`silent=true`**

Di seguito è riportato un esempio dello stesso comando di stato di cui sopra, tranne che il parametro `silent` è ora impostato su `true`:

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
  -d "silent=true"
```

Questo comando restituisce la stessa risposta del caso `silent=false`, ma la query annullata restituisce ora un codice HTTP 200 con una risposta simile alla seguente:

```
{
  "head" : {
    "vars" : [ "s", "p", "o" ]
  },
  "results" : {
    "bindings" : [ ]
  }
}
```

Utilizzo del protocollo HTTP Graph Store Protocol (GSP) SPARQL 1.1 in Amazon Neptune

Nella raccomandazione SPARQL 1.1 Graph Store HTTP Protocol, il W3C ha definito un protocollo HTTP per la gestione dei grafi RDF. Definisce le operazioni per la rimozione, la creazione e la sostituzione del contenuto del grafo RDF, nonché per l'aggiunta di istruzioni RDF al contenuto esistente.

Il protocollo GSP (Graph-Store Protocol) offre un modo pratico per manipolare l'intero grafo senza dover scrivere query SPARQL complesse.

A partire dal Rilascio: 1.0.5.0 (27/07/2021), Neptune supporta pienamente questo protocollo.

L'endpoint del protocollo GSP (Graph-Store Protocol) è:

```
https://your-neptune-cluster:port/sparql/gsp/
```

Per accedere al grafo predefinito con GSP, usare:

```
https://your-neptune-cluster:port/sparql/gsp/?default
```

Per accedere a un grafo nominato con GSP, usare:

```
https://your-neptune-cluster:port/sparql/gsp/?graph=named-graph-URI
```

Dettagli speciali sull'implementazione di Neptune GSP

Neptune implementa pienamente la raccomandazione W3C che definisce il protocollo GSP. Tuttavia, ci sono alcune situazioni che la specifica non copre.

Una di queste è il caso in cui una richiesta PUT o POST specifichi uno o più grafi nominati nel corpo della richiesta che differiscono dal grafo specificato dall'URL della richiesta. Ciò può accadere solo quando il formato RDF del corpo della richiesta supporta grafici nominati, come, ad esempio, con `Content-Type: application/n-quads` o `Content-Type: application/trig`.

In questa situazione, Neptune aggiunge o aggiorna tutti i grafi nominati presenti nel corpo, oltre al grafo nominato specificato nell'URL.

Ad esempio, supponiamo che partendo da un database vuoto, si invii una richiesta PUT per trasformare i voti in tre grafi. Uno, denominato `urn:votes`, contiene tutti i voti di tutti gli anni elettorali. Altri due, denominati `urn:votes:2005` e `urn:votes:2019`, contengono i voti di specifici anni elettorali. La richiesta e il relativo payload avranno un aspetto simile al seguente:

```
PUT "http://your-Neptune-cluster:port/sparql/gsp/?graph=urn:votes"
  Host: example.com
  Content-Type: application/n-quads

  PAYLOAD:

  <urn:JohnDoe> <urn:votedFor> <urn:Labour> <urn:votes:2005>
  <urn:JohnDoe> <urn:votedFor> <urn:Conservative> <urn:votes:2019>
  <urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes:2005>
  <urn:JaneSmith> <urn:votedFor> <urn:Conservative> <urn:votes:2019>
```

Dopo l'esecuzione della richiesta, i dati nel database avranno un aspetto simile al seguente:

```
<urn:JohnDoe>   <urn:votedFor> <urn:Labour>          <urn:votes:2005>
<urn:JohnDoe>   <urn:votedFor> <urn:Conservative>    <urn:votes:2019>
```

```
<urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes:2005>
<urn:JaneSmith> <urn:votedFor> <urn:Conservative>     <urn:votes:2019>
<urn:JohnDoe>   <urn:votedFor> <urn:Labour>           <urn:votes>
<urn:JohnDoe>   <urn:votedFor> <urn:Conservative>     <urn:votes>
<urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes>
<urn:JaneSmith> <urn:votedFor> <urn:Conservative>     <urn:votes>
```

Un'altra situazione ambigua è quella in cui più di un grafo viene specificato nell'URL della richiesta stesso, utilizzando uno qualsiasi di `PUT`, `POST`, `GET` o `DELETE`. Per esempio:

```
POST "http://your-Neptune-cluster:port/sparql/gsp/?
graph=urn:votes:2005&graph=urn:votes:2019"
```

O:

```
GET "http://your-Neptune-cluster:port/sparql/gsp/?default&graph=urn:votes:2019"
```

In questa situazione, Neptune restituisce un errore HTTP 400 con un messaggio che indica che è possibile specificare un solo grafo nell'URL della richiesta.

Analisi dell'esecuzione di query Neptune tramite la funzionalità SPARQL `explain`

Amazon Neptune ha aggiunto una funzionalità SPARQL denominata explain. Questa funzionalità è uno strumento self-service che consente di comprendere l'approccio di esecuzione adottato dal motore Neptune. È possibile richiamarla aggiungendo un parametro `explain` a una chiamata HTTP che invia una query SPARQL.

La funzione `explain` fornisce informazioni sulla struttura logica dei piani di esecuzione di query. È possibile utilizzare queste informazioni per individuare potenziali colli di bottiglia nella valutazione ed esecuzione. È quindi possibile utilizzare hint di query per migliorare i piani di esecuzione delle query.

Argomenti

- Funzionamento del motore di query SPARQL in Neptune
- Come utilizzare SPARQL explain per analizzare l'esecuzione di query Neptune
- Esempi di chiamate di SPARQL explain in Neptune
- Operatori di SPARQL explain in Neptune
- Limitazioni di SPARQL explain in Neptune

Funzionamento del motore di query SPARQL in Neptune

Per usare le informazioni fornite dalla funzionalità SPARQL `explain`, è necessario conoscere alcuni dettagli su come funziona il motore di query SPARQL di Amazon Neptune.

Il motore traduce ogni query SPARQL in una pipeline di operatori. A partire dal primo operatore, le soluzioni intermedie note come elenchi vincolanti passano attraverso questa pipeline di operatori. Si può considerare un elenco vincolante come una tabella in cui le intestazioni di tabella sono un sottoinsieme delle variabili utilizzate nella query. Ogni riga della tabella rappresenta un risultato, fino al punto di valutazione.

Supponiamo che siano stati definiti due prefissi di spazio dei nomi per i dati:

```
@prefix ex:   <http://example.com> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

Il seguente è un esempio di un elenco vincolante semplice in questo contesto:

```
?person       | ?firstName
------------------------------------------------------
ex:JaneDoe    | "Jane"
ex:JohnDoe    | "John"
ex:RichardRoe | "Richard"
```

Per ognuna delle tre persone, l'elenco associa la variabile `?person` a un identificatore della persona e la variabile `?firstName` al nome della persona.

In generale, le variabili possono rimanere non vincolate se, per esempio, c'è una selezione `OPTIONAL` di una variabile in una query per la quale non è presente nessun valore nei dati.

L'operatore `PipelineJoin` è un esempio di un operatore del motore di query Neptune presente nell'output di `explain`. Prende come input un set vincolante in entrata dall'operatore precedente ed esegue il join in base a un modello triplice, tipo `(?person, foaf:lastName, ?lastName)`. Questa operazione utilizza le associazioni per la variabile `?person` nel flusso di input, le sostituisce nel modello triplice e cerca le triple dal database.

Quando viene eseguito nel contesto delle associazioni in entrata dalla tabella precedente, `PipelineJoin` valuta tre ricerche, ovvero le seguenti:

```
(ex:JaneDoe,    foaf:lastName, ?lastName)
(ex:JohnDoe,    foaf:lastName, ?lastName)
```

```
  (ex:RichardRoe, foaf:lastName, ?lastName)
```

Questo approccio è definito valutazione as-bound. Le soluzioni di questo processo di valutazione verranno unite alle soluzioni in entrata, inserendo il `?lastName` rilevato nelle soluzioni in entrata. Supponendo che si trovi un cognome per tutte e tre le persone, l'operatore produrrebbe un elenco vincolante in uscita simile a questo:

```
?person       | ?firstName | ?lastName
---------------------------------------
ex:JaneDoe    | "Jane"     | "Doe"
ex:JohnDoe    | "John"     | "Doe"
ex:RichardRoe | "Richard"  | "Roe"
```

Questo elenco vincolante in uscita quindi serve come input per l'operatore successivo nella pipeline. Al termine, l'output dell'ultimo operatore nella pipeline definisce il risultato della query.

Le pipeline di operatori sono spesso lineari, nel senso che ogni operatore genera soluzioni per un unico operatore collegato. Tuttavia, in alcuni casi, possono avere più strutture complesse. Ad esempio, un operatore `UNION` in una query SPARQL viene mappato a un'operazione `Copy`. Questa operazione duplica le associazioni e inoltra le copie in due piani secondari, uno per il lato sinistro e l'altro per il lato destro di `UNION`.

Per ulteriori informazioni sugli operatori, vedi Operatori di SPARQL `explain` in Neptune.

Come utilizzare SPARQL **`explain`** per analizzare l'esecuzione di query Neptune

La funzionalità SPARQL `explain` è uno strumento self-service di Amazon Neptune che consente di comprendere l'approccio di esecuzione adottato dal motore Neptune. Per richiamare `explain`, si passa un parametro a una richiesta HTTP o HTTPS nel formato `explain=`*`mode`*.

Il valore della modalità può essere `static`, `dynamic` o `details`.

- In modalità statica, `explain` visualizza solo la struttura statica del piano di query.
- In modalità dinamica, `explain` include anche gli aspetti dinamici del piano di query. Questi aspetti potrebbero includere il numero di associazioni intermedie che passano attraverso gli operatori e il rapporto tra le associazioni in entrata e quelle in uscita e il tempo totale impiegato dagli operatori.
- Nella modalità dettagli `explain` stampa le informazioni visualizzate in modalità `dynamic` oltre a dettagli aggiuntivi, come la stringa di query SPARQL effettiva e il calcolo dell'intervallo stimato per il modello sottostante a un operatore join.

Neptune supporta l'utilizzo di `explain` con tutti e tre i protocolli di accesso alle query SPARQL elencati nella specifica W3C SPARQL 1.1 Protocol, ovvero:

1. HTTP GET
2. HTTP POST utilizzando i parametri di codifica URL
3. HTTP POST usando i parametri di testo

Per ulteriori informazioni sul motore di query SPARQL, consulta Funzionamento del motore di query SPARQL in Neptune.

Per informazioni sul tipo di output ottenuto richiamando SPARQL `explain`, consulta Esempi di chiamate di SPARQL `explain` in Neptune.

Esempi di chiamate di SPARQL **`explain`** in Neptune

Gli esempi di questa sezione mostrano i vari tipi di output che è possibile produrre richiamando la funzionalità SPARQL `explain` per analizzare l'esecuzione di query in Amazon Neptune.

Argomenti

- Descrizione dell'output di Explain
- Esempio di output con modalità dettagli
- Esempio di output con modalità statica
- Diversi metodi di codifica dei parametri
- Altri tipi di output oltre a text/plain
- Esempio di output di SPARQL explain quando è abilitato il motore DFE

Descrizione dell'output di Explain

In questo esempio, Jane Doe conosce due persone, ossia John Doe e Richard Roe:

```
@prefix ex: <http://example.com> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

ex:JaneDoe foaf:knows ex:JohnDoe .
ex:JohnDoe foaf:firstName "John" .
ex:JohnDoe foaf:lastName "Doe" .
```

```
ex:JaneDoe foaf:knows ex:RichardRoe .
ex:RichardRoe foaf:firstName "Richard" .
ex:RichardRoe foaf:lastName "Roe" .
.
```

Per determinare i nomi di tutte le persone che Jane Doe conosce, puoi scrivere la seguente query:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -H "Accept: text/csv"
```

Questa query semplice restituisce il seguente risultato:

```
firstName
John
Richard
```

Quindi, modifica il comando `curl` per richiamare `explain` aggiungendo `-d "explain=dynamic"` e utilizzando il tipo di output predefinito invece di `text/csv`:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=dynamic"
```

La query restituisce ora l'output in formato ASCII formattato (tipo di contenuto HTTP `text/plain`), che è il tipo di output predefinito:

```
########################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     # Units In # Units Out # Ratio # Time (ms) #
########################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        # 0         # 1          # 0.00  # 0         #
########################################################################################
```

```
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        # 1        # 2         # 2.00  # 1         #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person]
            #          #          #           #       #           #
#################################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        # 2        # 2         # 1.00  # 1         #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person, ?firstName]
            #          #          #           #       #           #
#################################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   # 2        # 2         # 1.00  # 0         #
#################################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value # 2        # 2         # 1.00  # 1         #
#################################################################################################
```

Per ulteriori informazioni sulle operazioni nella colonna `Name` e i relativi argomenti, consulta Operatori explain.

Di seguito viene descritto l'output riga per riga:

1. La prima fase della query principale utilizza sempre l'operatore `SolutionInjection` per introdurre una soluzione. La soluzione viene quindi ampliata per il risultato finale attraverso il processo di valutazione.

   In questo caso, introduce la cosiddetta soluzione universale `{ }`. In presenza di clausole `VALUES` o un `BIND`, questa fase potrebbe anche introdurre associazioni di variabili più complesse per iniziare.

   La colonna `Units Out` indica che questa è l'unica soluzione prodotta dall'operatore. La colonna `Out #1` specifica l'operatore in cui questo operatore immette il risultato. In questo esempio, tutti gli operatori sono collegati all'operatore che segue nella tabella.
2. La seconda fase è un `PipelineJoin`. Riceve come input la soluzione universale (completamente non vincolata) unica prodotta dall'operatore precedente (`Units In := 1`). Esegue il join in base al modello di tupla definito dall'argomento `pattern`. Questo corrisponde a una semplice ricerca del modello. In questo caso, il modello triplice è definito come segue:

```
distinct( ex:JaneDoe, foaf:knows, ?person )
```

L'argomento `joinType := join` indica che questo è un join normale (altri tipi includono join `optional`, `existence check` e così via).

L'argomento `distinct := true` indica che vengono estratte solo le corrispondenze distinte dal database (senza duplicati) e le corrispondenze distinte vengono associate alla variabile `joinProjectionVars := ?person`, deduplicate.

Il fatto che il valore della colonna `Units Out` sia 2 indica che ci sono due soluzioni in uscita. Nello specifico, queste sono le associazioni per la variabile `?person`, riflettendo le due persone che i dati indicano che Jane Doe conosce:

```
?person
-------------
ex:JohnDoe
ex:RichardRoe
```

3. Le due soluzioni dalla fase 2 passano come input (`Units In := 2`) al secondo `PipelineJoin`. Questo operatore esegue il join delle due soluzioni precedenti con il seguente modello triplice:

   ```
   distinct(?person, foaf:firstName, ?firstName)
   ```

   La variabile `?person` è associata a `ex:JohnDoe` o a `ex:RichardRoe` dalla soluzione in entrata dell'operatore. A questo punto, `PipelineJoin` estrae i nomi, John e Richard. Le due soluzioni in uscita (Units Out := 2) sono quindi le seguenti:

   ```
   ?person       | ?firstName
   --------------------------
   ex:JohnDoe    | John
   ex:RichardRoe | Richard
   ```

4. Il successivo operatore di proiezione prende come input le due soluzioni dalla fase 3 (`Units In := 2`) e le proietta alla variabile `?firstName`. Ciò elimina tutte le altre associazioni di variabili nelle mappature e inoltra le due associazioni (`Units Out := 2`):

   ```
   ?firstName
   ----------
   John
   ```

```
  Richard
```

5. Per migliorare le prestazioni, Neptune opera dove possibile con identificatori interni che assegna a termini quali URI e valori letterali di stringa, piuttosto che con le stringhe stesse. L'operatore finale, `TermResolution`, esegue una mappatura da questi identificatori interni verso le stringhe di termini corrispondenti.

   In una valutazione di query regolare (non Explain), il risultato calcolato dall'ultimo operatore viene quindi serializzato nel formato di serializzazione richiesto e trasmesso al client.

Esempio di output con modalità dettagli

> **Note**
>
> La modalità dettagli di SPARQL explain è disponibile a partire dal rilascio 1.0.2.1 del motore Neptune.

Si supponga di eseguire la stessa query precedente in modalità dettagli anziché in modalità dinamica:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=details"
```

Come illustrato in questo esempio, l'output è lo stesso con alcuni dettagli aggiuntivi quali la stringa di query nella parte superiore dell'output e il conteggio `patternEstimate` per l'operatore `PipelineJoin`:

```
Query:
PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://www.example.com/>
SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person foaf:firstName ?
firstName }

################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     # Units In # Units Out # Ratio # Time (ms) #
################################################################################################
```

```
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        # 0        # 1         # 0.00  # 0         #
##########################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        # 1        # 2         # 2.00  # 13        #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person]
            #          #          #           #       #           #
#    #        #        #                   # patternEstimate=2
            #          #          #           #       #           #
##########################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        # 2        # 2         # 1.00  # 3         #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person, ?firstName]
            #          #          #           #       #           #
#    #        #        #                   # patternEstimate=2
            #          #          #           #       #           #
##########################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   # 2        # 2         # 1.00  # 1         #
##########################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value # 2        # 2         # 1.00  # 7         #
##########################################################################################
```

Esempio di output con modalità statica

Supponiamo di eseguire la stessa query precedente in modalità statica (impostazione predefinita) anziché in modalità dettagli:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=static"
```

Come mostrato in questo esempio, l'output è lo stesso ma omette le ultime tre colonne:

```
##########################################################################################
```

```
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     #
#########################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        #
#########################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        #
#    #        #        #                   # joinType=join
            #          #
#    #        #        #                   # joinProjectionVars=[?person]
            #          #
#########################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        #
#    #        #        #                   # joinType=join
            #          #
#    #        #        #                   # joinProjectionVars=[?person, ?firstName]
            #          #
#########################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   #
#########################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value #
#########################################################################################
```

Diversi metodi di codifica dei parametri

Le query nell'esempio seguente illustrano due diversi metodi per codificare i parametri quando si richiama SPARQL `explain`.

Utilizzo di codifica URL: questo esempio utilizza la codifica URL dei parametri e specifica l'output dinamico:

```
curl -XGET "http(s)://your_server:your_port/sparql?query=SELECT%20*%20WHERE%20%7B
%20%3Fs%20%3Fp%20%3Fo%20%7D%20LIMIT%20%31&explain=dynamic"
```

Specifica diretta dei parametri: questa è la stessa query precedente con la differenza che trasferisce i parametri direttamente tramite POST:

```
curl http(s)://your_server:your_port/sparql \
  -d "query=SELECT * WHERE { ?s ?p ?o } LIMIT 1" \
```

```
  -d "explain=dynamic"
```

Altri tipi di output oltre a text/plain

Gli esempi precedenti utilizzano il tipo di output predefinito `text/plain`. Neptune può anche formattare l'output di SPARQL `explain` in altri due formati di tipo MIME, vale a dire `text/csv` e `text/html`. È possibile richiamarli impostando l'intestazione `Accept` HTTP. A tale scopo, si può utilizzare il flag `-H` in `curl`, come segue:

```
  -H "Accept: output type"
```

Ecco alcuni esempi:

Output **`text/csv`**

Questa query richiede l'output di tipo MIME CSV specificando `-H "Accept: text/csv"`:

```
curl http(s)://your_server:your_port/sparql \
  -d "query=SELECT * WHERE { ?s ?p ?o } LIMIT 1" \
  -d "explain=dynamic" \
  -H "Accept: text/csv"
```

Il formato CSV, che è utile per l'importazione in un foglio di calcolo o database, separa i campi in ogni riga `explain` con un punto e virgola (`;`), come segue:

```
ID;Out #1;Out #2;Name;Arguments;Mode;Units In;Units Out;Ratio;Time (ms)
0;1;-;SolutionInjection;solutions=[{}];-;0;1;0.00;0
1;2;-;PipelineJoin;pattern=distinct(?s, ?p, ?o),joinType=join,joinProjectionVars=[?s, ?
p, ?o];-;1;6;6.00;1
2;3;-;Projection;vars=[?s, ?p, ?o];retain;6;6;1.00;2
3;-;-;Slice;limit=1;-;1;1;1.00;1
```

Output **`text/html`**

Se si specifica `-H "Accept: text/html"`, `explain` genera una tabella HTML:

```
<!DOCTYPE html>
<html>
  <body>
    <table border="1px">
```

```
      <thead>
        <tr>
          <th>ID</th>
          <th>Out #1</th>
          <th>Out #2</th>
          <th>Name</th>
          <th>Arguments</th>
          <th>Mode</th>
          <th>Units In</th>
          <th>Units Out</th>
          <th>Ratio</th>
          <th>Time (ms)</th>
        </tr>
      </thead>

      <tbody>
        <tr>
          <td>0</td>
          <td>1</td>
          <td>-</td>
          <td>SolutionInjection</td>
          <td>solutions=[{}]</td>
          <td>-</td>
          <td>0</td>
          <td>1</td>
          <td>0.00</td>
          <td>0</td>
        </tr>

        <tr>
          <td>1</td>
          <td>2</td>
          <td>-</td>
          <td>PipelineJoin</td>
          <td>pattern=distinct(?s, ?p, ?o)<br>
              joinType=join<br>
              joinProjectionVars=[?s, ?p, ?o]</td>
          <td>-</td>
          <td>1</td>
          <td>6</td>
          <td>6.00</td>
          <td>1</td>
        </tr>
```

```
        <tr>
          <td>2</td>
          <td>3</td>
          <td>-</td>
          <td>Projection</td>
          <td>vars=[?s, ?p, ?o]</td>
          <td>retain</td>
          <td>6</td>
          <td>6</td>
          <td>1.00</td>
          <td>2</td>
        </tr>

        <tr>
          <td>3</td>
          <td>-</td>
          <td>-</td>
          <td>Slice</td>
          <td>limit=1</td>
          <td>-</td>
          <td>1</td>
          <td>1</td>
          <td>1.00</td>
          <td>1</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

L'HTML viene reso in un browser in modo simile al seguente:

ID	Out #1	Out #2	Name	Arguments	Mode	Units In	Units Out	Ratio	Time (ms)
0	1	-	SolutionInjection	solutions=[{}]	-	0	1	0.00	0
1	2	-	PipelineJoin	pattern=distinct(?s, ?p, ?o) joinType=join joinProjectionVars=[?s, ?p, ?o]	-	1	6	6.00	1
2	3	-	Projection	vars=[?s, ?p, ?o]	retain	6	6	1.00	2
3	-	-	Slice	limit=1	-	1	1	1.00	1

Esempio di output di SPARQL **`explain`** quando è abilitato il motore DFE

Di seguito è riportato un esempio di output di SPARQL `explain` quando è abilitato il motore di query alternativo DFE:

```
################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments


 # Mode     # Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]


 # -        # 0        # 1         # 0.00  # 0         #
################################################################################################
# 1  # 2      # -      # HashIndexBuild    # solutionSet=solutionSet1


 # -        # 1        # 1         # 1.00  # 22        #
#    #        #        #                   # joinVars=[]


 #          #          #           #       #           #
#    #        #        #                   # sourceType=pipeline


 #          #          #           #       #           #
################################################################################################
# 2  # 3      # -      # DFENode            # DFE Stats=


      # -        # 101      # 100       # 0.99  # 32        #
#    #        #        #                   # ====> DFE execution time (measured by
 DFEQueryEngine)

     #          #          #           #       #           #
#    #        #        #                   # accepted [micros]=127


 #          #          #           #       #           #
#    #        #        #                   # ready [micros]=2
```

```
 #          #          #           #       #           #
#    #        #        #                   # running [micros]=5627


 #          #          #           #       #           #
#    #        #        #                   # finished [micros]=0


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # ===> DFE execution time (measured in
 DFENode)

      #          #          #           #       #           #
#    #        #        #                   # -> setupTime [ms]=1


 #          #          #           #       #           #
#    #        #        #                   # -> executionTime [ms]=14


 #          #          #           #       #           #
#    #        #        #                   # -> resultReadTime [ms]=0


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # ===> Static analysis statistics
```

```
 #          #          #           #       #           #
#    #        #        #                   # --> 35907 micros spent in parser.


 #          #          #           #       #           #
#    #        #        #                   # --> 7643 micros spent in range count
 estimation

      #          #          #           #       #           #
#    #        #        #                   # --> 2895 micros spent in value resolution


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # --> 39974925 micros spent in optimizer
 loop

    #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # DFEJoinGroupNode[ children={


  #          #          #           #       #           #
#    #        #        #                   #   DFEPatternNode[(?1, TERM[117442062], ?
2, ?3) . project DISTINCT[?1, ?2] {rangeCountEstimate=100},

  #          #          #           #       #           #
#    #        #        #                   #     OperatorInfoWithAlternative[


  #          #          #           #       #           #
#    #        #        #                   #       rec=OperatorInfo[
```

```
  #          #           #           #       #           #
#    #        #        #                  #         type=INCREMENTAL_PIPELINE_JOIN,


 #          #           #           #       #           #
#    #        #        #                  #
 costEstimates=OperatorCostEstimates[

                                    #          #          #           #       #
  #
#    #        #        #                  #
 costEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0002,comp=0.0000,mem=0],

                                   #          #          #           #       #
 #
#    #        #        #                  #
 worstCaseCostEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0002,comp=0.0000,mem=0]

                      #          #          #           #       #           #
#    #        #        #                  #       alt=OperatorInfo[


  #          #           #           #       #           #
#    #        #        #                  #         type=INCREMENTAL_HASH_JOIN,


 #          #           #           #       #           #
#    #        #        #                  #
 costEstimates=OperatorCostEstimates[

                                    #          #          #           #       #
  #
#    #        #        #                  #
 costEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0003,comp=0.0000,mem=3212],

                               #          #          #           #       #           #
#    #        #        #                  #
 worstCaseCostEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0003,comp=0.0000,mem=32

                  #          #          #           #       #           #
#    #        #        #                  #   DFEPatternNode[(?1, TERM[150997262], ?
4, ?5) . project DISTINCT[?1, ?4] {rangeCountEstimate=100},
```

```
  #          #           #           #       #           #
#    #        #        #                  #     OperatorInfoWithAlternative[


  #          #           #           #       #           #
#    #        #        #                  #       rec=OperatorInfo[


  #          #           #           #       #           #
#    #        #        #                  #         type=INCREMENTAL_HASH_JOIN,


 #          #           #           #       #           #
#    #        #        #                  #
 costEstimates=OperatorCostEstimates[

                                   #          #          #           #       #
  #
#    #        #        #                  #
 costEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0003,comp=0.0000,mem=6400],

                              #          #          #           #       #           #
#    #        #        #                  #
 worstCaseCostEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0003,comp=0.0000,mem=

                 #          #          #           #       #           #
#    #        #        #                  #       alt=OperatorInfo[


  #          #           #           #       #           #
#    #        #        #                  #         type=INCREMENTAL_PIPELINE_JOIN,


 #          #           #           #       #           #
#    #        #        #                  #
 costEstimates=OperatorCostEstimates[

                                   #          #          #           #       #
  #
#    #        #        #                  #
 costEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0010,comp=0.0000,mem=0],

                                #          #          #           #       #           #
```

```
#    #        #        #                   #
 worstCaseCostEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0010,comp=0.0000,mem=

                   #          #          #           #       #           #
#    #        #        #                   # },


 #          #          #           #       #           #
#    #        #        #                   # ]


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # ===> DFE configuration:


 #          #          #           #       #           #
#    #        #        #                   # solutionChunkSize=5000


 #          #          #           #       #           #
#    #        #        #                   # ouputQueueSize=20


 #          #          #           #       #           #
#    #        #        #                   # numComputeCores=3


 #          #          #           #       #           #
#    #        #        #                   # maxParallelIO=10


 #          #          #           #       #           #
#    #        #        #                   # numInitialPermits=12


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
```

```
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # ====> DFE configuration (reported back)


 #          #          #           #       #           #
#    #        #        #                  # numComputeCores=3


 #          #          #           #       #           #
#    #        #        #                  # maxParallelIO=2


 #          #          #           #       #           #
#    #        #        #                  # numInitialPermits=12


 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # ===> Statistics & operator histogram


 #          #          #           #       #           #
#    #        #        #                  # ==> Statistics


 #          #          #           #       #           #
#    #        #        #                  # -> 3741 / 3668 micros total elapsed (incl.
 wait / excl. wait)

 #          #          #           #       #           #
#    #        #        #                  # -> 3741 / 3 millis total elapse (incl.
 wait / excl. wait)

    #          #          #           #       #           #
#    #        #        #                  # -> 3741 / 0 secs total elapsed (incl.
 wait / excl. wait)

     #          #          #           #       #           #
```

```
#    #        #        #                  # ==> Operator histogram


 #          #          #           #       #           #
#    #        #        #                  # -> 47.66% of total time (excl. wait):
 pipelineScan (2 instances)

     #          #          #           #       #           #
#    #        #        #                  # -> 10.99% of total time (excl. wait):
 merge (1 instances)

     #          #          #           #       #           #
#    #        #        #                  # -> 41.17% of total time (excl. wait):
 symmetricHashJoin (1 instances)

     #          #          #           #       #           #
#    #        #        #                  # -> 0.19% of total time (excl. wait): drain
 (1 instances)

 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # nodeId | out0   | out1 | opName
   | args                                          | rowsIn | rowsOut | chunksIn |
 chunksOut | elapsed* | outWait | outBlocked | ratio    | rate* [M/s] | rate [M/s] | %
    #          #          #           #       #           #
#    #        #        #                  # ------ | ------ | ---- | -----------------
 | ---------------------------------------------- | ------ | ------- | -------- |
 --------- | -------- | ------- | ---------- | -------- | ----------- | ---------- |
 ----- #          #          #           #       #           #
#    #        #        #                  # node_0 | node_2 | -    | pipelineScan
  | (?1, TERM[117442062], ?2, ?3) DISTINCT [?1, ?2]  | 0      | 100     | 0        | 1
        | 874      | 0       | 0          | Infinity | 0.1144      | 0.1144     | 23.83
 #          #          #           #       #           #
#    #        #        #                  # node_1 | node_2 | -    | pipelineScan
  | (?1, TERM[150997262], ?4, ?5) DISTINCT [?1, ?4]  | 0      | 100     | 0        | 1
        | 874      | 0       | 0          | Infinity | 0.1144      | 0.1144     | 23.83
 #          #          #           #       #           #
#    #        #        #                  # node_2 | node_4 | -    | symmetricHashJoin
 |                                                 | 200    | 100     | 2        | 2
        | 1510     | 73      | 0          | 0.50     | 0.0662      | 0.0632     | 41.17
 #          #          #           #       #           #
```

```
#    #        #        #                  # node_3 | -      | -    | drain
  |                                                | 100    | 0       | 1        | 0
        | 7        | 0       | 0          | 0.00     | 0.0000      | 0.0000     | 0.19
 #          #          #          #       #          #
#    #        #        #                  # node_4 | node_3 | -    | merge
  |                                                | 100    | 100     | 2        | 1
        | 403      | 0       | 0          | 1.00     | 0.2481      | 0.2481     | 10.99
 #          #          #          #       #          #
#########################################################################################
# 3  # 4      # -      # HashIndexJoin     # solutionSet=solutionSet1


 # -        # 100      # 100       # 1.00  # 4         #
#    #        #        #                  # joinType=join


 #          #          #          #       #          #
#########################################################################################
# 4  # 5      # -      # Distinct          # vars=[?s, ?o, ?o1]


 # -        # 100      # 100       # 1.00  # 9         #
#########################################################################################
# 5  # 6      # -      # Projection        # vars=[?s, ?o, ?o1]


 # retain   # 100      # 100       # 1.00  # 2         #
#########################################################################################
# 6  # -      # -      # TermResolution    # vars=[?s, ?o, ?o1]


 # id2value # 100      # 100       # 1.00  # 11        #
#########################################################################################
```

Operatori di SPARQL **`explain`** in Neptune

Le seguenti sezioni descrivono gli operatori e i parametri per la funzionalità SPARQL `explain` attualmente disponibile in Amazon Neptune.

> **Important**
>
> La funzione SPARQL `explain` è ancora in fase di definizione. Gli operatori e i parametri documentati qui potrebbero cambiare nelle prossime versioni.

Argomenti

Operatore **`Aggregation`**

Esegue una o più aggregazioni, implementando la semantica di operatori di aggregazione SPARQL, come `count`, `max`, `min`, `sum` e così via.

`Aggregation` viene fornito con il raggruppamento opzionale utilizzando le clausole `groupBy` e i vincoli opzionali `having`.

Argomenti

- `groupBy`: (facoltativo) fornisce una clausola `groupBy` che specifica la sequenza di espressioni in base a cui le soluzioni in entrata sono raggruppate.
- `aggregates`: (obbligatorio) specifica un elenco ordinato di espressioni di aggregazione.
- `having`: (facoltativo) aggiunge vincoli per filtrare gruppi, come implicito nella clausola `having` della query SPARQL.

Operatore **`ConditionalRouting`**

Instrada le soluzioni in entrata in base a una determinata condizione. Le soluzioni che soddisfano la condizione vengono instradate all'ID operatore a cui fa riferimento `Out #1`, mentre le soluzioni che non la soddisfano vengono instradate all'operatore a cui fa riferimento `Out #2`.

Argomenti

- `condition`: (obbligatorio) la condizione di instradamento.

Operatore **`Copy`**

Delega il flusso di soluzione come specificato dalla modalità indicata.

Modalità

- `forward`: inoltra le soluzioni all'operatore downstream identificato da `Out #1`.
- `duplicate`: duplica le soluzioni e le inoltra a ciascuno dei due operatori identificati da `Out #1` e `Out #2`.

`Copy` non ha argomenti.

Operatore **`DFENode`**

Questo operatore è un'astrazione del piano eseguito dal motore di query alternativo DFE. Il piano DFE dettagliato è illustrato negli argomenti di questo operatore. L'argomento è attualmente sovraccarico per contenere le statistiche dettagliate di runtime del piano DFE. Contiene il tempo impiegato nei vari passaggi di esecuzione delle query da parte del motore DFE.

L'albero sintattico astratto (AST) logico ottimizzato per il piano di query DFE viene stampato con informazioni sui tipi di operatori presi in considerazione durante la pianificazione e sui costi associati nel caso peggiore e nel caso migliore per l'esecuzione degli operatori. Al momento, l'AST è composto dai seguenti tipi di nodi:

- `DFEJoinGroupNode`: rappresenta un join di uno o più `DFEPatternNodes`.
- `DFEPatternNode`: incapsula un modello sottostante con cui le tuple corrispondenti vengono proiettate fuori dal database sottostante.

La sottosezione `Statistics & Operator histogram` contiene dettagli sul tempo di esecuzione del piano `DataflowOp` e sulla ripartizione del tempo di CPU utilizzato da ciascun operatore. Di seguito è riportata una tabella che riporta le statistiche dettagliate di runtime del piano eseguito dal motore DFE.

> **Note**
>
> Poiché il motore DFE è una funzionalità sperimentale rilasciata in modalità di laboratorio, il formato esatto dell'output di `explain` potrebbe cambiare.

Operatore **`Distinct`**

Calcola la proiezione Distinct su un sottoinsieme di variabili, eliminando i duplicati. Di conseguenza, il numero di soluzioni in entrata è maggiore o uguale al numero di soluzioni in uscita.

Argomenti

- `vars`: (obbligatorio) variabili a cui si applica la proiezione `Distinct`.

Operatore **`Federation`**

Passa una determinata query a un endpoint SPARQL remoto specifico.

Argomenti

- `endpoint`: (obbligatorio) URL dell'endpoint nella dichiarazione SPARQL `SERVICE`. Può essere una stringa costante oppure, se l'endpoint della query è determinato in base a una variabile all'interno della stessa query, il nome della variabile.

- `query`: (obbligatorio) stringa di query ricostruita da inviare all'endpoint remoto. Il motore aggiunge prefissi predefiniti a questa query anche quando non vengono specificati dal client.
- `silent`: (obbligatorio) valore booleano che indica se la parola chiave `SILENT` è visualizzata dopo la parola chiave. `SILENT` indica al motore di non considerare tutta la query non riuscita anche se la parte `SERVICE` remota non riesce.

Operatore **`Filter`**

Filtra le soluzioni in entrata. Solo le soluzioni che soddisfano la condizione di filtro vengono inoltrate all'operatore upstream, mentre le altre vengono rilasciate.

Argomenti

- `condition`: (obbligatorio) condizione di filtro.

Operatore **`HashIndexBuild`**

Prende un elenco di associazioni e le elabora in un indice hash il cui nome viene definito dall'argomento `solutionSet`. Di solito, gli operatori successivi eseguono join in base a questo set di soluzioni, facendovi riferimento con quel nome.

Argomenti

- `solutionSet`: (obbligatorio) nome del set di soluzioni dell'indice hash.
- `sourceType`: (obbligatorio) tipo di origine da cui vengono ottenute le associazioni da archiviare nell'indice hash.
  - `pipeline`: elabora le soluzioni in entrata dall'operatore downstream nella pipeline dell'operatore nell'indice hash.
  - `binding set`: elabora il set di associazioni fisse specificate dall'argomento `sourceBindingSet` nell'indice hash.
- `sourceBindingSet`: (facoltativo) se il valore dell'argomento `sourceType` è `binding set`, questo argomento specifica il set di associazioni statico da elaborare nell'indice hash.

Operatore **`HashIndexJoin`**

Esegue il join delle soluzioni in entrata rispetto al set di soluzioni dell'indice hash identificato dall'argomento `solutionSet`.

Argomenti

- `solutionSet`: (obbligatorio) nome del set di soluzioni rispetto al quale eseguire il join. Questo deve essere un indice hash che è stato costruito in una fase precedente utilizzando l'operatore `HashIndexBuild`.
- `joinType`: (obbligatorio) tipo di join da eseguire.
  - `join`: join normale che richiede una corrispondenza esatta tra tutte le variabili condivise.
  - `optional`: join `optional` che usa la semantica dell'operatore SPARQL `OPTIONAL`.
  - `minus`: un'operazione `minus` conserva una mappatura per la quale non esiste alcun partner di join, utilizzando la semantica dell'operatore SPARQL `MINUS`.
  - `existence check`: verifica se c'è un partner di join o meno e associa la variabile `existenceCheckResultVar` al risultato di questo controllo.
- `constraints`: (facoltativo) vincoli di join aggiuntivi considerati durante il join. I join che non soddisfano questi vincoli vengono eliminati.
- `existenceCheckResultVar`: (facoltativo) usato solo per i join in cui `joinType` è uguale a `existence check` (vedi l'argomento `joinType` precedente).

Operatore **`MergeJoin`**

Un merge join su più set di soluzioni, identificati dall'argomento `solutionSets`.

Argomenti

- `solutionSets`: (obbligatorio) set di soluzioni su cui eseguire il join.

Operatore **`NamedSubquery`**

Attiva la valutazione della sottoquery identificata dall'argomento `subQuery` ed elabora il risultato nel set di soluzioni specificato dall'argomento `solutionSet`. Le soluzioni in entrata per l'operatore sono inoltrate alla sottoquery e quindi all'operatore successivo.

Argomenti

- `subQuery`: (obbligatorio) nome della sottoquery da valutare. La sottoquery viene resa in modo esplicito nell'output.
- `solutionSet`: (obbligatorio) nome del set di soluzioni in cui archiviare il risultato della sottoquery.

Operatore **PipelineJoin**

Riceve come input l'output dell'operatore precedente ed esegue il join in base al modello di tupla definito dall'argomento `pattern`.

Argomenti

- `pattern`— (Obbligatorio) Il pattern, che assume la forma di una tupla subject-predicate-object, e facoltativamente di un grafico, che sta alla base del join. Se `distinct` viene specificato per il modello, il join estrae solo le soluzioni distinte dalle variabili di proiezione specificate dall'argomento `projectionVars`, piuttosto che tutte le soluzioni corrispondenti.
- `inlineFilters`: (facoltativo) set di filtri da applicare alle variabili nel modello. Il modello viene valutato in combinazione con questi filtri.
- `joinType`: (obbligatorio) tipo di join da eseguire.
  - `join`: join normale che richiede una corrispondenza esatta tra tutte le variabili condivise.
  - `optional`: join `optional` che usa la semantica dell'operatore SPARQL `OPTIONAL`.
  - `minus`: un'operazione `minus` conserva una mappatura per la quale non esiste alcun partner di join, utilizzando la semantica dell'operatore SPARQL `MINUS`.
  - `existence check`: verifica se c'è un partner di join o meno e associa la variabile `existenceCheckResultVar` al risultato di questo controllo.
- `constraints`: (facoltativo) vincoli di join aggiuntivi considerati durante il join. I join che non soddisfano questi vincoli vengono eliminati.
- `projectionVars`: (facoltativo) variabili della proiezione. Utilizzato in combinazione con `distinct := true` per applicare l'estrazione delle proiezioni distinte per un determinato set di variabili.
- `cutoffLimit`: (facoltativo) limite massimo per il numero di partner di join estratti. Anche se non vi è alcun limite predefinito, è possibile impostare questo a 1 quando si eseguono join per implementare clausole `FILTER (NOT) EXISTS`, dove è sufficiente provare o smentire che c'è un partner di join.

Operatore **PipelineCountJoin**

Variante di `PipelineJoin`. Invece di eseguire il join, conteggia solo i partner di join corrispondenti e associa il conteggio alla variabile specificata dall'argomento `countVar`.

Argomenti

- `countVar`: (obbligatorio) variabile a cui il risultato del conteggio, cioè il numero di partner di join, deve essere associato.
- `pattern`— (Obbligatorio) Il pattern, che assume la forma di una tupla subject-predicate-object, e facoltativamente a grafo, che sta alla base dell'unione. Se `distinct` viene specificato per il modello, il join estrae solo le soluzioni distinte dalle variabili di proiezione specificate dall'argomento `projectionVars`, piuttosto che tutte le soluzioni corrispondenti.
- `inlineFilters`: (facoltativo) set di filtri da applicare alle variabili nel modello. Il modello viene valutato in combinazione con questi filtri.
- `joinType`: (obbligatorio) tipo di join da eseguire.
  - `join`: join normale che richiede una corrispondenza esatta tra tutte le variabili condivise.
  - `optional`: join `optional` che usa la semantica dell'operatore SPARQL `OPTIONAL`.
  - `minus`: un'operazione `minus` conserva una mappatura per la quale non esiste alcun partner di join, utilizzando la semantica dell'operatore SPARQL `MINUS`.
  - `existence check`: verifica se c'è un partner di join o meno e associa la variabile `existenceCheckResultVar` al risultato di questo controllo.
- `constraints`: (facoltativo) vincoli di join aggiuntivi considerati durante il join. I join che non soddisfano questi vincoli vengono eliminati.
- `projectionVars`: (facoltativo) variabili della proiezione. Utilizzato in combinazione con `distinct := true` per applicare l'estrazione delle proiezioni distinte per un determinato set di variabili.
- `cutoffLimit`: (facoltativo) limite massimo per il numero di partner di join estratti. Anche se non vi è alcun limite predefinito, è possibile impostare questo a 1 quando si eseguono join per implementare clausole `FILTER (NOT) EXISTS`, dove è sufficiente provare o smentire che c'è un partner di join.

Operatore **`PipelinedHashIndexJoin`**

Si tratta di un operatore di all-in-one build hash index e join. Accetta un elenco di associazioni, le elabora in un indice hash e quindi esegue il join delle soluzioni in entrata rispetto all'indice hash.

Argomenti

- `sourceType`: (obbligatorio) tipo di origine da cui vengono ottenute le associazioni da archiviare nell'indice hash, uno tra:

  - `pipeline`: fa sì che `PipelinedHashIndexJoin` elabori le soluzioni in entrata dall'operatore downstream nella pipeline dell'operatore nell'indice hash.
  - `binding set`: fa sì che `PipelinedHashIndexJoin` elabori il set di associazioni fisse specificate dall'argomento `sourceBindingSet` nell'indice hash.
- `sourceSubQuery` : (facoltativo) se il valore dell'argomento `sourceType` è `pipeline`, questo argomento specifica la sottoquery che viene valutata ed elaborata nell'indice hash.
- `sourceBindingSet` : (facoltativo) se il valore dell'argomento `sourceType` è `binding set`, questo argomento specifica il set di associazioni statico da elaborare nell'indice hash.
- `joinType`: (obbligatorio) tipo di join da eseguire:
  - `join`: join normale che richiede una corrispondenza esatta tra tutte le variabili condivise.
  - `optional`: join `optional` che usa la semantica dell'operatore SPARQL `OPTIONAL`.
  - `minus`: un'operazione `minus` conserva una mappatura per la quale non esiste alcun partner di join, utilizzando la semantica dell'operatore SPARQL `MINUS`.
  - `existence check`: verifica se c'è un partner di join o meno e associa la variabile `existenceCheckResultVar` al risultato di questo controllo.
- `existenceCheckResultVar`: (facoltativo) usato solo per i join in cui `joinType` è uguale a `existence check` (vedi l'argomento joinType precedente).

Operatore **`Projection`**

Proietta su un sottoinsieme di variabili. Il numero di soluzioni in entrata è uguale al numero di soluzioni in uscita, ma la forma della soluzione varia in base all'impostazione della modalità.

Modalità

- `retain`: conserva nelle soluzioni solo le variabili specificate dall'argomento `vars`.
- `drop`: elimina tutte le variabili specificate dall'argomento `vars`.

Argomenti

- `vars`: (obbligatorio) variabili da conservare o eliminare, a seconda dell'impostazione della modalità.

Operatore **`PropertyPath`**

Abilita percorsi di proprietà ricorsivi come `+` o `*`. Neptune implementa un approccio di iterazione a virgola fissa in base a un modello specificato dall'argomento `iterationTemplate`. Le variabili lato destro o lato sinistro note sono vincolate nel modello per ogni iterazione a virgola fissa, fino a quando non vengono più trovate nuove soluzioni.

Argomenti

- `iterationTemplate`: (obbligatorio) nome del modello di sottoquery utilizzato per implementare l'iterazione a virgola fissa.
- `leftTerm`: (obbligatorio) termine (variabile o costante) sul lato sinistro del percorso di proprietà.
- `rightTerm`: (obbligatorio) termine (variabile o costante) sul lato destro del percorso di proprietà.
- `lowerBound`: (obbligatorio) limite inferiore per l'iterazione a virgola fissa (`0` per le query `*` oppure `1` per le query `+`).

Operatore **`TermResolution`**

Traduce i valori dell'identificatore della stringa interna nelle stringhe esterni corrispondenti o traduce le stringhe esterne in valori dell'identificatore della stringa interna, in base alla modalità.

Modalità

- `value2id`: mappa termini come valori letterali e URI ai valori di ID interni corrispondenti (codifica in valori interni).
- `id2value`: mappa i valori di ID interni ai termini corrispondenti come valori letterali e URI (decodifica di valori interni).

Argomenti

- `vars`: (obbligatorio) specifica le variabili le cui stringhe o ID di stringa interna devono essere mappate.

Operatore **`Slice`**

Implementa una sezione nel flusso di soluzioni in entrata, utilizzando la semantica delle clausole SPARQL `LIMIT` e `OFFSET`.

Argomenti

- `limit`: (facoltativo) limite per le soluzioni da inoltrare.
- `offset`: (facoltativo) offset a cui sono valutate le soluzioni per l'inoltro.

Operatore **`SolutionInjection`**

Non riceve alcun input. Introduce staticamente le soluzioni nel piano di query e le registra nell'argomento `solutions`.

I piani di query iniziano sempre con questa introduzione statica. Se le soluzioni statiche da introdurre possono essere ricavate dalla query stessa combinando varie origini di associazioni statiche (ad esempio da clausole `VALUES` o `BIND`), l'operatore `SolutionInjection` introduce queste soluzioni statiche derivate. Nel caso più semplice, queste riflettono associazioni che sono implicite in una clausola `VALUES` esterna.

Se nessuna delle soluzioni statiche può essere ricavata dalla query, `SolutionInjection` introduce la soluzione vuota cosiddetta universale, che viene ampliata e moltiplicata durante tutto il processo di valutazione di query.

Argomenti

- `solutions`: (obbligatorio) sequenza di soluzioni introdotte dall'operatore.

Operatore **`Sort`**

Ordina il set di soluzioni utilizzando le condizioni di ordinamento specificate.

Argomenti

- `sortOrder`: (obbligatorio) elenco ordinato di variabili, ognuna contenente un identificatore `ASC` (crescente) o `DESC` (decrescente), utilizzate sequenzialmente per ordinare il set di soluzioni.

Operatore **`VariableAlignment`**

Analizza le soluzioni una ad una, eseguendo l'allineamento su ognuna in base a due variabili: un `sourceVar` specificato e un `targetVar` specificato.

Se `sourceVar` e `targetVar` in una soluzione hanno lo stesso valore, le variabili sono considerate allineate e la soluzione viene inoltrata, con il `sourceVar` ridondante proiettato in uscita.

Se le variabili sono associate a diversi valori, la soluzione viene completamente filtrata.

Argomenti

- `sourceVar`: (obbligatorio) variabile di origine da confrontare con la variabile di destinazione. Se l'allineamento va a buon fine in una soluzione, il che significa che le due variabili hanno lo stesso valore, la variabile di origine viene proiettata in uscita.
- `targetVar`: (obbligatorio) variabile di destinazione con la quale viene confrontata la variabile di origine. Viene conservata anche quando l'allineamento va a buon fine.

Limitazioni di SPARQL **`explain`** in Neptune

Il rilascio della funzionalità SPARQL `explain` in Neptune ha i seguenti limiti.

Neptune attualmente supporta explain solo in query SPARQL SELECT

Per informazioni sul processo di valutazione per altri formati di query, ad esempio le query `ASK`, `CONSTRUCT`, `DESCRIBE` e `SPARQL UPDATE`, è possibile trasformare queste query in una query SELECT. Quindi utilizzare `explain` per controllare la query SELECT corrispondente.

Ad esempio, per ottenere informazioni di `explain` su una query `ASK WHERE {...}`, eseguire la query corrispondente `SELECT WHERE {...} LIMIT 1` con `explain`.

Analogamente, per una query `CONSTRUCT {...} WHERE {...}`, eliminare la parte `CONSTRUCT {...}` ed eseguire una query `SELECT` con `explain` nella seconda clausola `WHERE {...}`. La valutazione della seconda clausola `WHERE` generalmente rivela le principali sfide per l'elaborazione della query `CONSTRUCT` poiché le soluzioni in uscita dal secondo `WHERE` nel modello `CONSTRUCT` generalmente richiedono solo una semplice sostituzione.

Gli operatori Explain possono cambiare nelle versioni future

Gli operatori SPARQL `explain` e i relativi parametri possono cambiare nelle versioni future

L'output di Explain può cambiare nelle versioni future

Ad esempio, potrebbero essere modificate le intestazioni di colonna e altre colonne potrebbero essere aggiunte alle tabelle.

Query federate SPARQL in Neptune che utilizzano l'estensione `SERVICE`

Amazon Neptune supporta completamente l'estensione delle query federate SPARQL che utilizza la parola chiave `SERVICE`. Per ulteriori informazioni, consulta Query federate SPARQL 1.1.

> **Note**
>
> Questa funzionalità è disponibile a partire dal Rilascio 1.0.1.0.200463.0 (15/10/2019).

La parola chiave `SERVICE` indica al motore di query SPARQL di eseguire una parte della query rispetto a un endpoint SPARQL remoto e di comporre il risultato della query finale. Sono possibili solo le operazioni `READ`. Le operazioni `WRITE` e `DELETE` non sono supportate. Neptune può eseguire solo query federate su endpoint SPARQL accessibili all'interno del proprio cloud privato virtuale (VPC). Tuttavia, è anche possibile utilizzare un proxy inverso nel VPC per rendere accessibile un'origine dati esterna all'interno del VPC.

> **Note**
>
> Quando SPARQL `SERVICE` viene utilizzato per federare una query a due o più cluster Neptune nello stesso VPC, i gruppi di sicurezza devono essere configurati per consentire a tutti i cluster Neptune di parlare tra loro.

> **Important**
>
> La federazione SPARQL 1.1 effettua richieste di servizio per tuo conto quando trasferisci query e parametri a endpoint SPARQL esterni. È tua responsabilità verificare che gli endpoint SPARQL esterni soddisfino i requisiti di sicurezza e gestione dei dati dell'applicazione.

Esempio di una query federata Neptune

Il seguente esempio semplice mostra il funzionamento di query federate SPARQL.

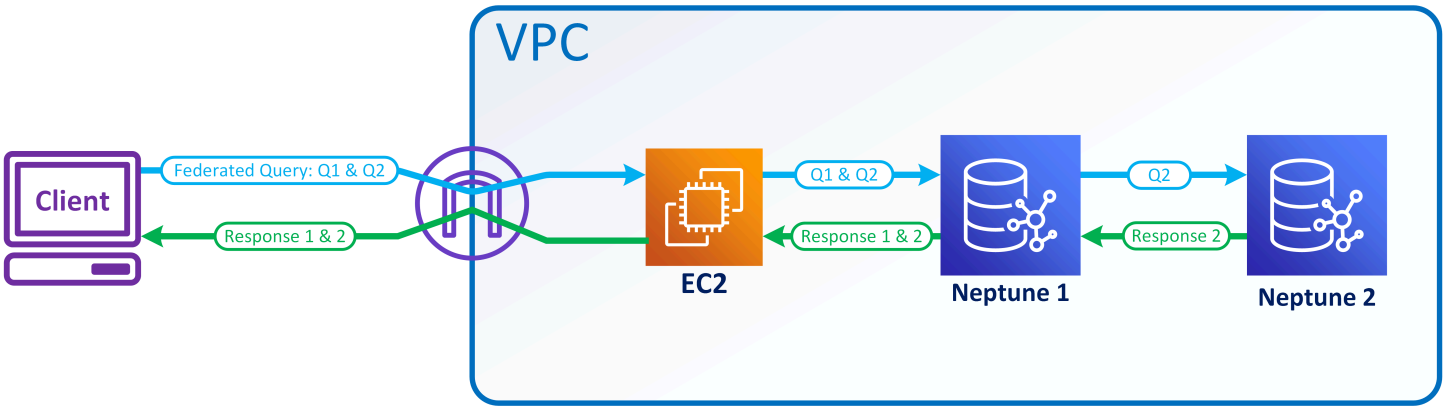
Supponiamo che un cliente invii la seguente query a Neptune-1 all'indirizzo `http://neptune-1:8182/sparql`.

```
SELECT * WHERE {
```

```
    ?person rdf:type foaf:Person .
    SERVICE <http://neptune-2:8182/sparql> {
        ?person foaf:knows ?friend .
     }
}
```

1. Neptune-1 valuta il primo modello di query (Q-1) che è `?person rdf:type foaf:Person`, utilizza i risultati per risolvere `?person` in Q-2 (`?person foaf:knows ?friend`) e inoltra il modello risultante a Neptune-2 all'indirizzo `http://neptune-2:8182/sparql`.
2. Neptune-2 valuta Q-2 e restituisce i risultati a Neptune-1.
3. Neptune-1 unisce le soluzioni per entrambi i modelli e restituisce i risultati al cliente.

Questo flusso è mostrato nel seguente diagramma.



> **Note**
>
> Per impostazione predefinita, il sistema di ottimizzazione determina a che punto dell'esecuzione della query viene eseguita l'istruzione `SERVICE`. È possibile sostituire questo posizionamento utilizzando l'hint di query joinOrder.

Controllo degli accessi per query federate in Neptune

Neptune AWS Identity and Access Management utilizza (IAM) per l'autenticazione e l'autorizzazione. Il controllo degli accessi per una query federata può riguardare più istanze database Neptune. Queste istanze potrebbero avere requisiti diversi per il controllo degli accessi. In alcune circostanze, questo può limitare la possibilità di effettuare una query federata.

Consideriamo il semplice esempio presentato nella sezione precedente. Neptune-1 chiama Neptune-2 con le stesse credenziali con cui è stato chiamato.

- Se Neptune-1 richiede l'autenticazione e l'autorizzazione IAM, ma Neptune-2 non le richiede, per effettuare la query federata occorrono solo le autorizzazioni IAM appropriate per Neptune-1.
- Se Neptune-1 e Neptune-2 richiedono entrambi l'autenticazione e l'autorizzazione IAM, è necessario collegare le autorizzazioni IAM per entrambi i database per effettuare la query federata.
- Tuttavia, nel caso in cui Neptune-1 non sia abilitato per IAM ma Neptune-2 lo sia, non è possibile effettuare una query federata. Il motivo è che Neptune-1 non può recuperare le credenziali IAM e passarle a Neptune-2 per autorizzare la seconda parte della query.

Strumenti di visualizzazione dei grafi per Neptune

Oltre alle funzionalità di visualizzazione integrate nei notebook grafici Neptune, puoi anche utilizzare soluzioni create AWS da partner e fornitori terzi per visualizzare i dati archiviati in Neptune.

Una visualizzazione sofisticata dei grafi consente ai data scientist, ai responsabili e ad altri ruoli di un'organizzazione di esplorare i dati dei grafi in modo interattivo, senza dover sapere come scrivere query complesse.

Argomenti

- Graph-explorer open source
- Tom Sawyer Software
- Cambridge Intelligence
- Graphistry
- metaphacts
- G.V( )
- Linkurious

Graph-explorer open source

Graph-explorer è uno strumento di esplorazione visiva open source con poco codice per i dati dei grafi, disponibile con la licenza Apache-2.0. Consente di esplorare i grafi di proprietà etichettati (LPG) o i dati Resource Description Framework (RDF) in un database a grafo senza dover scrivere query sui grafi. Graph-explorer ha lo scopo di aiutare i data scientist, gli analisti aziendali e altri ruoli di un'organizzazione a esplorare i dati dei grafi in modo interattivo senza dover imparare un linguaggio di query a grafo.

Graph-explorer fornisce un'applicazione Web basata su React che può essere implementata come container per visualizzare i dati a grafo. Puoi connetterti ad Amazon Neptune o ad altri database grafici che forniscono un endpoint TinkerPop Apache Gremlin o SPARQL 1.1.

- È possibile visualizzare rapidamente un riepilogo dei dati utilizzando i filtri con facet o cercare i dati digitando il testo nella barra di ricerca.

- È possibile anche esplorare in modo interattivo le connessioni tra nodi e archi. È possibile visualizzare i nodi adiacenti per vedere come gli oggetti si relazionano tra loro e quindi eseguire il drill-down per ispezionare visivamente gli archi e le proprietà.
- È inoltre possibile personalizzare il layout del grafo, i colori, le icone e le proprietà predefinite da visualizzare per nodi e archi. Per i grafi RDF, è possibile personalizzare anche gli spazi dei nomi per gli URI delle risorse.
- Per i report e le presentazioni che coinvolgono i dati dei grafi, è possibile configurare e salvare le viste create in un formato PNG ad alta risoluzione. È anche possibile scaricare i dati associati in un file CSV o JSON per un'ulteriore elaborazione.

Utilizzo di graph-explorer in un notebook grafico Neptune

Il modo più semplice per usare graph-explorer con Neptune è in un notebook grafico Neptune.

Se si utilizza Neptune Workbench per ospitare un notebook Neptune, graph-explorer viene distribuito automaticamente con il notebook e connesso a Neptune.

Dopo aver creato un notebook, passa alla console Neptune per avviare graph-explorer:

1. Passa a Neptune.
2. In Notebook, seleziona il notebook.
3. In Azioni, scegli Apri Graph Explorer.

Come eseguire graph-explorer in Amazon ECS su AWS Fargate e connettersi a Neptune

Puoi anche creare l'immagine Docker di graph-explorer ed eseguirla su un computer locale o su un servizio ospitato come Amazon Elastic Compute Cloud (Amazon EC2) o Amazon Elastic Container Service (Amazon ECS), come spiegato nella sezione Getting Started del progetto read-me in the graph-explorer. GitHub

Ad esempio, questa sezione fornisce step-by-step istruzioni per eseguire graph-explorer in Amazon ECS su: AWS Fargate

1. Creare un nuovo ruolo IAM e collegarvi queste policy:
   - Amazonecs TaskExecutionRolePolicy

- CloudWatchLogsFullAccess

Tenere il nome del ruolo a portata di mano per utilizzarlo tra un minuto.

2. Creare un cluster Amazon ECS con l'infrastruttura impostata su FARGATE e le seguenti opzioni di rete:

   - VPC: impostare sul VPC in cui si trova il database Neptune.
   - `Subnets`: impostare sulle sottoreti pubbliche di tale VPC (rimuovere tutte le altre).

3. Creare una nuova definizione di attività JSON come segue:

```
{
  "family": "explorer-test",
  "containerDefinitions": [
    {
      "name": "graph-explorer",
      "image": "public.ecr.aws/neptune/graph-explorer:latest",
      "cpu": 0,
      "portMappings": [
        {
          "name": "graph-explorer-80-tcp",
          "containerPort": 80,
          "hostPort": 80,
          "protocol": "tcp",
          "appProtocol": "http"
        },
        {
          "name": "graph-explorer-443-tcp",
          "containerPort": 443,
          "hostPort": 443,
          "protocol": "tcp",
          "appProtocol": "http"
        }
      ],
      "essential": true,
      "environment": [
        {
          "name": "HOST",
          "value": "localhost"
        }
      ],
      "mountPoints": [],
```

```
      "volumesFrom": [],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "true",
          "awslogs-group": "/ecs/graph-explorer",
          "awslogs-region": "{region}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    }
  ],
  "taskRoleArn": "arn:aws:iam::{account_no}:role/{role_name_from_step_1}",
  "executionRoleArn": "arn:aws:iam::{account_no}:role/{role_name_from_step_1}",
  "networkMode": "awsvpc",
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "cpu": "1024",
  "memory": "3072",
  "runtimePlatform": {
    "cpuArchitecture": "X86_64",
    "operatingSystemFamily": "LINUX"
  }
}
```

4. Iniziare una nuova attività utilizzando le impostazioni predefinite, ad eccezione dei seguenti campi:

   - Ambiente
     - Opzioni di calcolo => Tipo di avvio
   - Configurazione della distribuzione
     - Tipo di applicazione => Attività
     - Famiglia => *(la nuova definizione di attività JSON*)
     - Revisione => *(la più recente)*
   - Reti
     - VPC => *(il VPC Neptune a cui connettersi)*
     - Sottoreti => *(SOLO le sottoreti pubbliche del VPC - rimuovere tutte le altre)*

- Gruppo di sicurezza => Crea un nuovo gruppo di sicurezza
- Nome del gruppo di sicurezza => graph-explorer
- Descrizione del gruppo di sicurezza = Gruppo di sicurezza per l'accesso a graph-explorer
- Regole in entrata per i gruppi di sicurezza =>
  1. 80 Anywhere
  2. 443 Anywhere

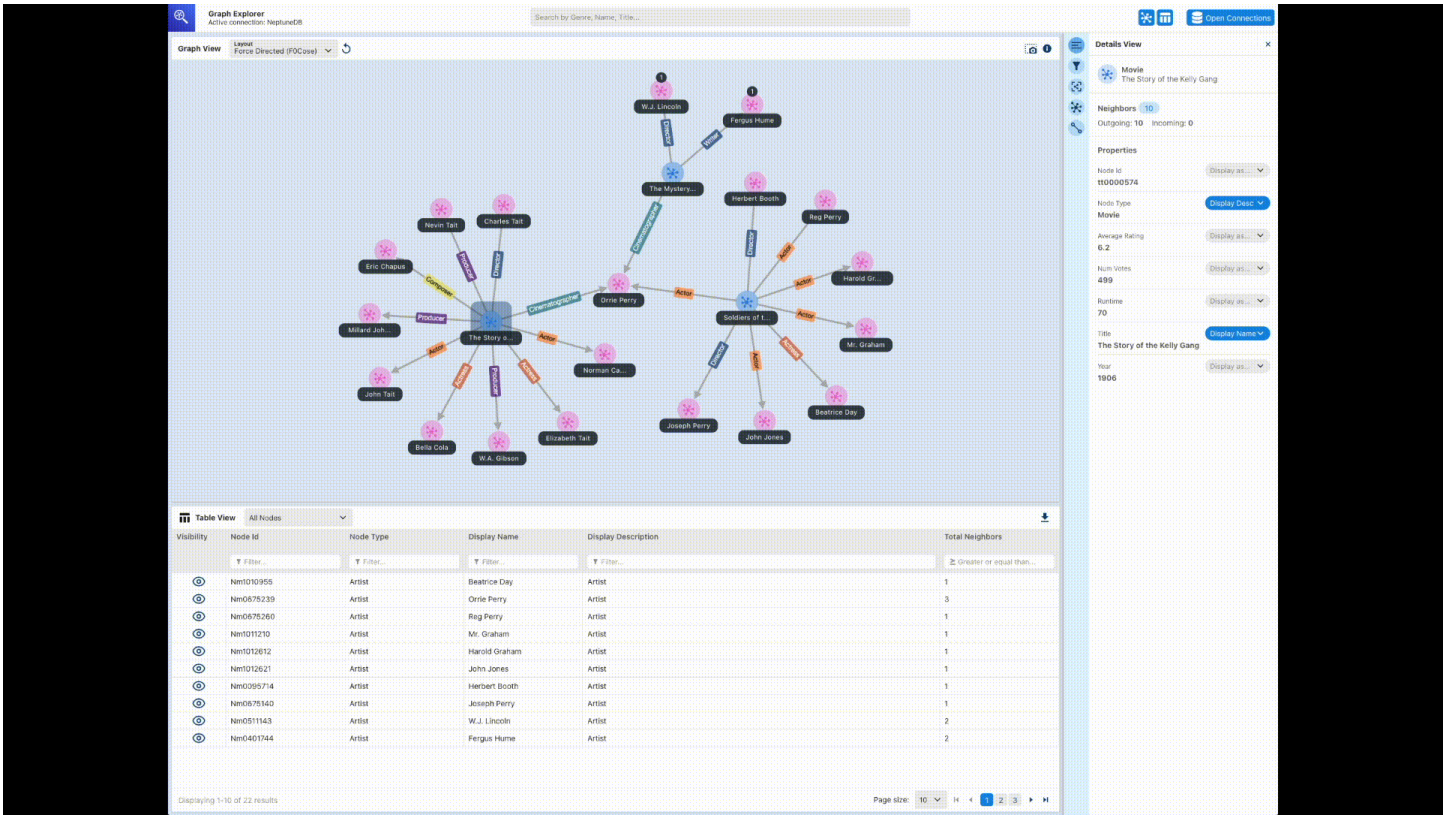
5. Seleziona Crea.
6. Dopo l'avvio dell'attività, copiare l'IP pubblico dell'attività in esecuzione e passare a: `https://(your public IP)/explorer`.
7. Accettare il rischio di utilizzare il certificato non riconosciuto che è stato generato o aggiungerlo al proprio keychain.
8. Ora è possibile aggiungere una connessione a Neptune. Creare una nuova connessione, per un grafo delle proprietà (GPL) o per RDF, e impostare i seguenti campi:

```
Using proxy server => true
Public or Proxy Endpoint => https://(your public IP address)
Graph connection URL => https://(your Neptune endpoint):8182
```

A questo punto si dovrebbe essere connessi.

Dimostrazione di graph-explorer

Questo breve video dà un'idea di come sia possibile visualizzare facilmente i dati del grafo utilizzando graph-explorer:



Tom Sawyer Software

Tom Sawyer Perspectives è una piattaforma di sviluppo di visualizzazione e analisi di grafi e dati con poco codice per i dati archiviati in Amazon Neptune. Le interfacce integrate di progettazione e anteprima e le ampie librerie API consentono di creare rapidamente applicazioni di visualizzazione personalizzate con qualità di produzione. Con un'interfaccia di point-and-click progettazione e 30 algoritmi di analisi integrati, puoi progettare e sviluppare applicazioni per ottenere informazioni dettagliate sui dati federati da dozzine di fonti.

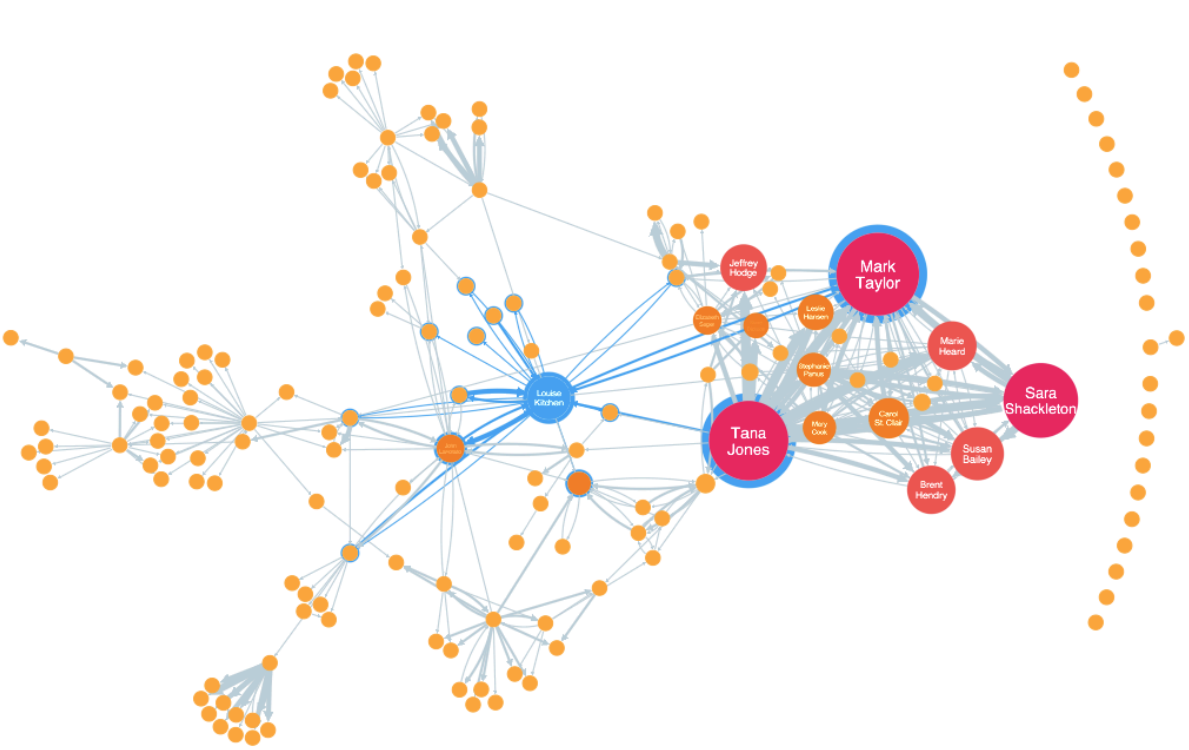
Tom Sawyer Graph Database Browser semplifica la visualizzazione e l'analisi dei dati in Amazon Neptune. Puoi vedere e comprendere le connessioni nei dati senza una conoscenza approfondita del linguaggio o dello schema di query. Puoi interagire con i dati senza conoscenze tecniche semplicemente caricando i nodi adiacenti ai nodi selezionati e costruendo la visualizzazione nella direzione desiderata. Puoi anche sfruttare cinque layout univoci per visualizzare il grafo nel modo più significativo e applicare analisi di centralità, clustering e pathfinding per rivelare modelli mai visti prima. Per vedere un esempio di integrazione di Graph Database Browser con Neptune, consulta questo post di blog. Per iniziare con una prova gratuita di Graph Database Browser, visita il Marketplace AWS.

Cambridge Intelligence

Cambridge Intelligence fornisce tecnologie di visualizzazione dei dati per esplorare e comprendere i dati di Amazon Neptune. I toolkit per la visualizzazione dei grafici (KeyLinesper JavaScript sviluppatori e sviluppatori React) offrono un modo semplice ReGraphper creare strumenti altamente interattivi e personalizzabili per applicazioni web. Questi toolkit sfruttano WebGL e HTML5 Canvas per prestazioni veloci, supportano funzioni avanzate di analisi dei grafi e combinano flessibilità e scalabilità con un'architettura sicura e affidabile. Questi SDK funzionano sia con i dati di Neptune Gremlin che con quelli RDF.

Consulta questi tutorial di integrazione per i dati Gremlin, i dati SPARQL e l'architettura Neptune.

Ecco un esempio di visualizzazione: KeyLines

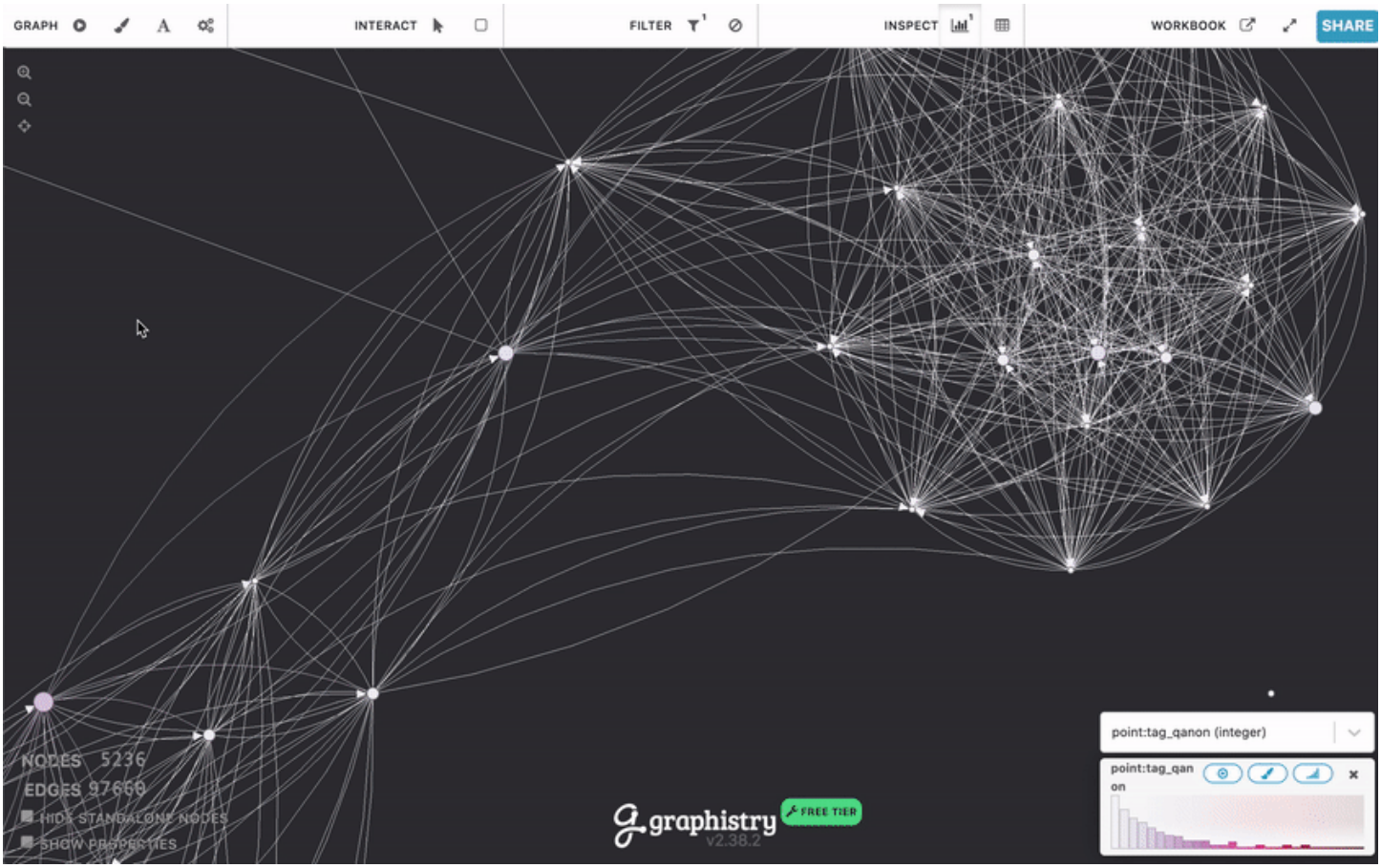


Graphistry

Graphistry è una piattaforma di intelligenza visiva basata su grafi che sfrutta l'accelerazione GPU per esperienze visive avanzate. I team possono collaborare su Graphistry utilizzando una varietà di funzionalità, dall'esplorazione senza codice di file e database, alla condivisione di notebook Jupyter e dashboard Streamlit, all'utilizzo dell'API di incorporamento nelle proprie app.

Puoi iniziare a utilizzare dashboard completamente interattivi a bassa codifica semplicemente configurando e avviando graph-app-kite modificando solo poche righe di codice. Consulta questo post di blog per una guida dettagliata sulla creazione della prima dashboard utilizzando Graphistry e Neptune. Puoi anche provare la demo di PyGraphistryNeptune. PyGraphistry è una libreria di analisi grafica visiva in Python per notebook. Dai un'occhiata a questo taccuino tutorial per una demo di PyGraphistry Neptune.

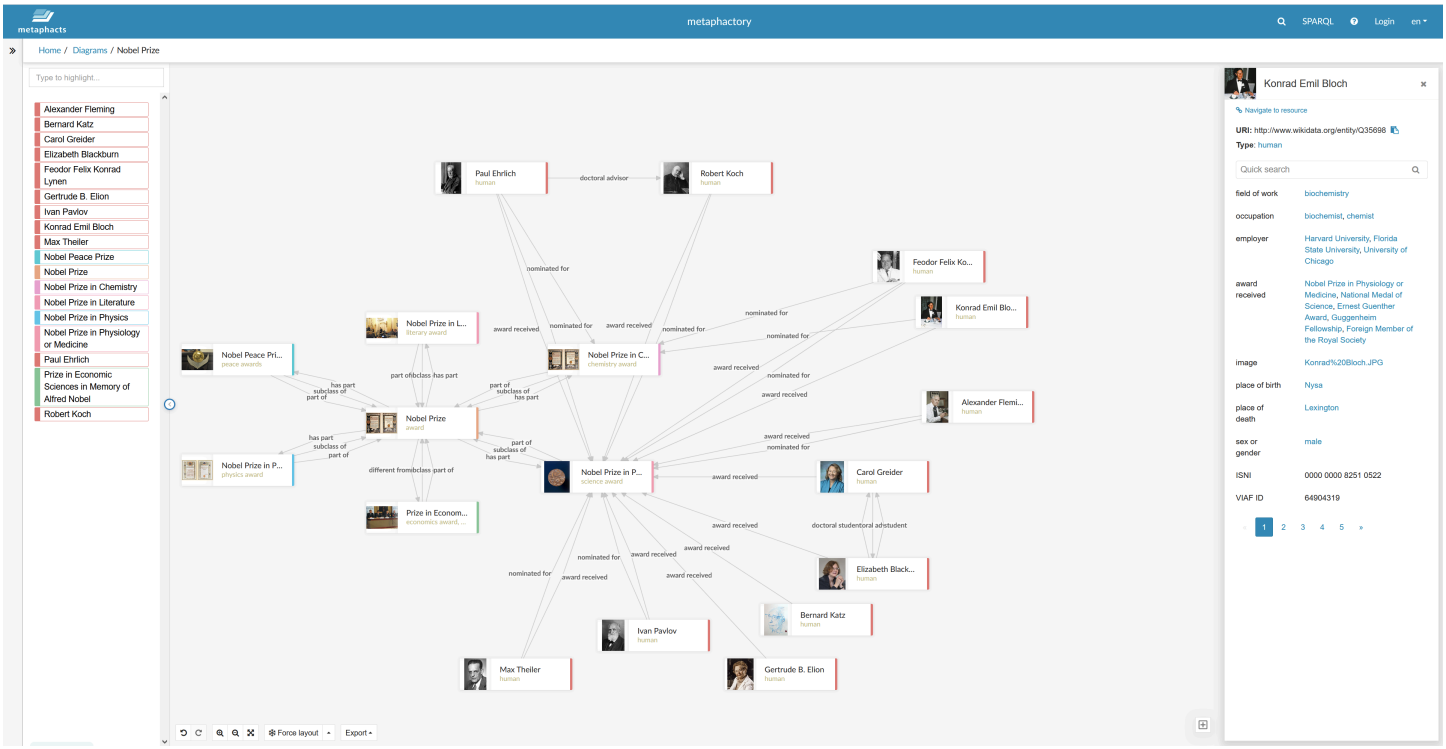
Per iniziare, visita Graphistry nel Marketplace AWS.



metaphacts

metaphacts offre una piattaforma flessibile e aperta per la descrizione e l'esecuzione di query sui dati dei grafi e per la visualizzazione e l'interazione con i grafi della conoscenza. Con metaphactory, è possibile creare applicazioni Web interattive come visualizzazioni e dashboard sulla base dei grafi della conoscenza di Neptune utilizzando il modello di dati RDF. La piattaforma metaphactory supporta un'esperienza di sviluppo con poco codice con un'interfaccia utente per il caricamento dei dati, un'interfaccia di modellazione visiva di rappresentazione formale con supporto OWL e SHACL, un'interfaccia utente e un catalogo di query SPARQL e un ampio set di componenti Web per l'esplorazione, la visualizzazione, la ricerca e la creazione di grafi.

Di seguito è riportata una visualizzazione di esempio di metaphactory:



La piattaforma è stata progettata ed è utilizzata in modo produttivo nei settori dell'ingegneria, della produzione, della farmaceutica, delle scienze della vita, della finanza, delle assicurazioni e altro ancora. Per vedere un esempio di architettura della soluzione, consulta questo post di blog.

Per iniziare con una prova gratuita di metaphactory, visita il Marketplace AWS.

G.V( )

G.V() è uno strumento dell'ambiente di sviluppo integrato (IDE) Gremlin per sviluppatori e analisti di dati. Con questo strumento puoi eseguire query, visualizzare e aggiornare in modo interattivo i dati dei grafi in Neptune. Puoi scrivere query utilizzando la funzionalità di completamento automatico integrata di G.V(), che offre suggerimenti e documentazione in base al modello di dati a grafo. Puoi anche utilizzare la funzione di debug delle query di Gremlin per scrivere, eseguire il debug, testare e analizzare in modo approfondito i processi di attraversamento dei grafi.

Con Data Explorer di G.V(), puoi navigare e modificare un grafo sia per creare rapidamente nuove strutture di grafi sia per mantenere quelle esistenti. G.V() offre diversi formati di visualizzazione per i risultati delle query, che consentono di interpretare l'output delle query e di navigare nel grafo in modo interattivo. Questi formati includono tabelle, grafi, JSON e formati di output della console Gremlin.

Per iniziare con una prova gratuita di G.V(), visita il sito web di G.V(), dove puoi anche ottenere ulteriori informazioni consultando la documentazione.

Di seguito è riportata una visualizzazione G.V() di esempio:

Linkurious

Linkurious offre diverse soluzioni di intelligenza grafica per utenti tecnici e non tecnici e una varietà di casi d'uso.

Linkurious Enterprise Explorer è un software di visualizzazione e analisi dei off-the-shelf grafici creato per i team in grado di stare al passo con le esigenze delle day-to-day attività e aiuta i professionisti orientati ai dati a fare grandi cose in modo semplice. Completamente configurabile e facile da usare, si adatta facilmente alle tue esigenze e consente ai principianti o agli utenti esperti di visualizzare rapidamente i dati in AWS Neptune, di esplorare in modo intuitivo il set di dati indipendentemente dalle dimensioni o dalla complessità dei dati e di collaborare senza problemi a livello di team o aziendale.

Linkurious Enterprise Watchtower sfrutta la potenza di Linkurious Enterprise Explorer e aggiunge funzionalità innovative di rilevamento e gestione dei casi per offrire un software integrato di rilevamento e indagine basato sulla tecnologia dei grafici. Da un lato, consente di configurare avvisi che sfruttano Neptune Database e Neptune Analytics per far emergere automaticamente anomalie o modelli in dati connessi complessi. Dall'altro, combina funzionalità di gestione dei casi e collaborazione per aiutare i team a gestire in modo efficiente i flussi di lavoro investigativi.

Ogma è una JavaScript libreria commerciale che ti aiuta a sviluppare visualizzazioni grafiche interattive potenti e su larga scala per le tue applicazioni. Sfrutta il rendering WebGL e i layout ad alte prestazioni per consentire agli utenti di visualizzare e interagire con migliaia di nodi e bordi in pochi secondi. Fornisce inoltre una varietà di funzionalità per personalizzare l'applicazione e creare esperienze utente avanzate. Infine, è dotato di documentazione e strumenti completi come tutorial, dozzine di esempi e un parco giochi interattivo.

Per iniziare, richiedi una prova gratuita di 30 giorni di Linkurious Enterprise o Ogma.

Esportazione di dati da un cluster database Neptune

Esistono diversi modi validi per esportare i dati da un cluster database Neptune:

- Per piccole quantità di dati, è sufficiente utilizzare i risultati di una o più query.
- Per i dati RDF, il protocollo GSP (Graph Store Protocol) semplifica l'esportazione. Ad esempio:

```
curl --request GET \
  'https://your-neptune-endpoint:port/sparql/gsp/?graph=http%3A//www.example.com/named/graph'
```

- Esiste anche uno strumento open source potente e flessibile per esportare i dati di Neptune, ovvero `neptune-export`. Le sezioni seguenti descrivono le funzionalità di questo strumento e spiegano come utilizzarle.

Argomenti

- Uso di neptune-export
- Utilizzo del servizio Neptune-Export per esportare i dati di Neptune
- Utilizzo dello strumento da riga di comando neptune-export per esportare i dati da Neptune
- File esportati da Neptune-Export e neptune-export
- Parametri utilizzati per controllare il processo di esportazione di Neptune
- Risoluzione dei problemi relativi al processo di esportazione di Neptune

Uso di **neptune-export**

È possibile utilizzare lo strumento `neptune-export` open source in due modi diversi:

- Come servizio Neptune-Export. Quando si esportano dati da Neptune utilizzando il servizio Neptune-Export, si attivano e si monitorano i processi di esportazione tramite una REST API.
- Come utilità da riga di comando Java **`neptune-export`**. Per utilizzare questo strumento da riga di comando per esportare i dati di Neptune, è necessario eseguirlo in un ambiente in cui il cluster database Neptune sia accessibile.

Sia il servizio Neptune-Export che lo strumento da riga di comando `neptune-export` pubblicano dati su Amazon Simple Storage Service (Amazon S3), crittografati tramite la crittografia lato server di Amazon S3 (`SSE-S3`).

> **Note**
>
> È consigliabile abilitare la registrazione degli accessi su tutti i bucket Amazon S3, per consentire l'audit di tutti gli accessi a tali bucket.

Se si tenta di esportare dati da un cluster database Neptune i cui dati vengono modificati durante l'esportazione, la coerenza dei dati esportati non è garantita. In altre parole, se il cluster gestisce il traffico di scrittura mentre è in corso un processo di esportazione, potrebbero verificarsi incoerenze nei dati esportati. Questo vale quando l'esportazione viene eseguita dall'istanza primaria del cluster o da una o più repliche di lettura.

Per garantire la coerenza dei dati esportati, è consigliabile esportare da un clone del cluster database. In questo modo si fornisce allo strumento di esportazione una versione statica dei dati e si garantisce che il processo di esportazione non rallenti le query nel cluster database originale.

Per semplificare questa operazione, puoi indicare che desideri clonare il cluster database di origine quando attivi un processo di esportazione. In tal caso, il processo di esportazione crea automaticamente il clone, lo utilizza per l'esportazione e quindi lo elimina al termine dell'esportazione.

Utilizzo del servizio Neptune-Export per esportare i dati di Neptune

Puoi utilizzare la procedura seguente per esportare dati dal tuo cluster database Neptune ad Amazon S3 con il servizio Neptune-Export:

Installazione del servizio Neptune-Export

Utilizza il modello AWS CloudFormation per creare lo stack:

Per installare il servizio Neptune-Export

1. Per avviare lo stack AWS CloudFormation nella console AWS CloudFormation, scegli uno dei pulsanti Avvia lo stack nella seguente tabella.

Region	Vista	Visualizzazione in Designer	Avvia
Stati Uniti orientali (Virginia settentrionale)	Visualizzazione	Visualizzazione in Designer	
Stati Uniti orientali (Ohio)	Visualizzazione	Visualizzazione in Designer	
Stati Uniti occidentali (California settentrionale)	Visualizzazione	Visualizzazione in Designer	
Stati Uniti occidentali (Oregon)	Visualizzazione	Visualizzazione in Designer	
Canada (Centrale)	Visualizzazione	Visualizzazione in Designer	
Sud America (San Paolo)	Visualizzazione	Visualizzazione in Designer	
Europa (Stoccolma)	Visualizzazione	Visualizzazione in Designer	

Region	Vista	Visualizzazione in Designer	Avvia
Europa (Irlanda)	Visualizzazione	Visualizzazione in Designer	
Europa (Londra)	Visualizzazione	Visualizzazione in Designer	
Europa (Parigi)	Visualizzazione	Visualizzazione in Designer	
Europa (Francoforte)	Visualizzazione	Visualizzazione in Designer	
Medio Oriente (Bahrein)	Visualizzazione	Visualizzazione in Designer	
Medio Oriente (Emirati Arabi Uniti)	Visualizzazione	Visualizzazione in Designer	
Israele (Tel Aviv)	Visualizzazione	Visualizzazione in Designer	
Africa (Città del Capo)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Hong Kong)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Tokyo)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Seul)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Singapore)	Visualizzazione	Visualizzazione in Designer	

Region	Vista	Visualizzazione in Designer	Avvia
Asia Pacifico (Sydney)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Mumbai)	Visualizzazione	Visualizzazione in Designer	
Cina (Pechino)	Visualizzazione	Visualizzazione in Designer	
Cina (Ningxia)	Visualizzazione	Visualizzazione in Designer	
AWS GovCloud (Stati Uniti occidentali)	Visualizzazione	Visualizzazione in Designer	
AWS GovCloud (Stati Uniti orientali)	Visualizzazione	Visualizzazione in Designer	

2. Nella pagina Select Template (Seleziona modello), selezionare Next (Avanti).
3. Nella pagina Specifica i dettagli del modello imposta i parametri seguenti:

   - **`VPC`**: il modo più semplice per configurare il servizio Neptune-Export è installarlo nello stesso VPC Amazon del database Neptune. Se desideri installarlo in un VPC separato, puoi utilizzare il peering VPC per stabilire la connettività tra il VPC del cluster database Neptune e il VPC del servizio Neptune-Export.
   - **`Subnet1`**: il servizio Neptune-Export deve essere installato in una sottorete del VPC che consenta il traffico HTTPS IPv4 in uscita dalla sottorete a Internet. In questo modo il servizio Neptune-Export può chiamare l'API Batch AWS per creare ed eseguire un processo di esportazione.

     Se hai creato il tuo cluster Neptune utilizzando il modello CloudFormation nella pagina Creazione di un cluster DB della documentazione di Neptune, puoi usare gli output `PrivateSubnet2` e `PrivateSubnet1` di tale stack per popolare questo parametro e quello successivo.

- **`Subnet2`**: una seconda sottorete nel VPC che consenta il traffico HTTPS IPv4 in uscita dalla sottorete a Internet.
- **`EnableIAM`**: imposta questo parametro su `true` per proteggere l'API Neptune-Endpoint con AWS Identity and Access Management (IAM). Questa impostazione è consigliata.

  Se abiliti l'autenticazione IAM, devi firmare tramite `Sigv4` tutte le richieste HTTPS all'endpoint. Puoi utilizzare uno strumento come awscurl per firmare le richieste per tuo conto.
- **`VPCOnly`**: impostando questo parametro su `true` si rende l'endpoint di esportazione solo VPC, affinché sia possibile accedervi solo dall'interno del VPC in cui è installato il servizio Neptune-Export. In questo modo, si limita l'utilizzo dell'API Neptune-Export solo all'interno di tale VPC.

  Consigliamo di impostare `VPCOnly` su `true`.
- **`NumOfFilesULimit`** : specifica un valore compreso tra 10.000 e 1.000.000 per `nofile` nella proprietà del container `ulimits`. Il valore predefinito è 10.000 ed è consigliabile lasciarlo invariato a meno che il grafo non contenga un numero elevato di etichette univoche.
- **`PrivateDnsEnabled`** (booleano): indica se associare o meno una zona ospitata privata al VPC specificato. Il valore predefinito è `true`.

  Quando viene creato un endpoint VPC con questo flag abilitato, tutto il traffico Gateway API viene instradato attraverso l'endpoint VPC e le chiamate all'endpoint Gateway API pubblico vengono disabilitate. Se si imposta `PrivateDnsEnabled` su `false`, l'endpoint Gateway API pubblico è abilitato, ma il servizio di esportazione Neptune non può essere connesso tramite l'endpoint DNS privato. È quindi possibile utilizzare un endpoint DNS pubblico per l'endpoint VPC per chiamare il servizio di esportazione, come descritto qui.

4. Seleziona Successivo.
5. Nella pagina Opzioni, scegli Avanti.
6. Nella pagina Revisione, seleziona la prima casella di controllo per accettare la creazione delle risorse IAM da parte di AWS CloudFormation. Seleziona la seconda casella di controllo per confermare `CAPABILITY_AUTO_EXPAND` per il nuovo stack.

> **Note**
>
> `CAPABILITY_AUTO_EXPAND` conferma in modo esplicito che, durante la creazione dello stack, le macro verranno ampliate senza revisione preventiva. Gli utenti spesso creano un set di modifiche da un modello elaborato, quindi le modifiche apportate dalle

macro possono essere riesaminate prima dell'effettiva creazione dello stack. Per ulteriori informazioni, consulta l'API CreateStack di AWS CloudFormation.

Quindi, scegli Crea.

Abilitazione dell'accesso a Neptune da Neptune-Export

Una volta completata l'installazione di Neptune-Export, aggiorna il gruppo di sicurezza VPC di Neptune per consentire l'accesso da Neptune-Export. Una volta creato lo stack AWS CloudFormation di Neptune-Export, la scheda Output include un ID `NeptuneExportSecurityGroup`. Aggiorna il tuo gruppo di sicurezza VPC di Neptune per consentire l'accesso da questo gruppo di sicurezza Neptune-Export.

Abilitazione dell'accesso all'endpoint Neptune-Export da un'istanza EC2 basata su VPC

Se imposti l'endpoint Neptune-Export come solo VPC, puoi accedervi solo dall'interno del VPC in cui è installato il servizio Neptune-Export. Per consentire la connettività da un'istanza Amazon EC2 nel VPC da cui è possibile effettuare chiamate API Neptune-Export, collega il gruppo di sicurezza `NeptuneExportSecurityGroup` creato dallo stack AWS CloudFormation a tale istanza Amazon EC2.

Esecuzione di un processo Neptune-Export con l'API Neptune-Export

La scheda Output dello stack AWS CloudFormation include anche `NeptuneExportApiUri`. Usa questo URI ogni volta che invii una richiesta all'endpoint Neptune-Export.

Esecuzione di un processo di esportazione

- Assicurati che all'utente o al ruolo in cui viene eseguita l'esportazione sia stata concessa l'autorizzazione `execute-api:Invoke`.
- Se hai impostato il parametro `EnableIAM` su `true` nello stack AWS CloudFormation quando hai installato Neptune-Export, devi `Sigv4` firmare tutte le richieste all'API Neptune-Export. Ti consigliamo di utilizzare awscurl per inviare le richieste all'API. Tutti gli esempi riportati qui presuppongono che l'autenticazione IAM sia abilitata.

- Se hai impostato il parametro `VPCOnly` su `true` nello stack AWS CloudFormation durante l'installazione di Neptune-Export, devi chiamare l'API Neptune-Export dall'interno del VPC, in genere da un'istanza Amazon EC2 disponibile nel VPC.

Per iniziare a esportare i dati, invia una richiesta all'endpoint `NeptuneExportApiUri` con i parametri di richiesta `command` e `outputS3Path`e un parametro di esportazione `endpoint`.

Quello che segue è un esempio di richiesta che esporta i dati del grafo delle proprietà da Neptune e li pubblica in Amazon S3:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-pg",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": { "endpoint": "(your Neptune endpoint DNS name)" }
      }'
```

Analogamente, ecco un esempio di richiesta che esporta i dati RDF da Neptune ad Amazon S3:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-rdf",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": { "endpoint": "(your Neptune endpoint DNS name)" }
      }'
```

Se si omette il parametro della richiesta `command`, per impostazione predefinita Neptune-Export tenta di esportare i dati del grafo delle proprietà da Neptune.

Se il comando precedente è stato eseguito correttamente, l'output avrà un aspetto simile al seguente:

```
{
  "jobName": "neptune-export-abc12345-1589808577790",
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f"
```

```
}
```

Monitoraggio del processo di esportazione appena avviato

Per monitorare un processo in esecuzione, aggiungi il relativo jobID a `NeptuneExportApiUri`, in modo simile al seguente:

```
curl \
  (your NeptuneExportApiUri)(the job ID)
```

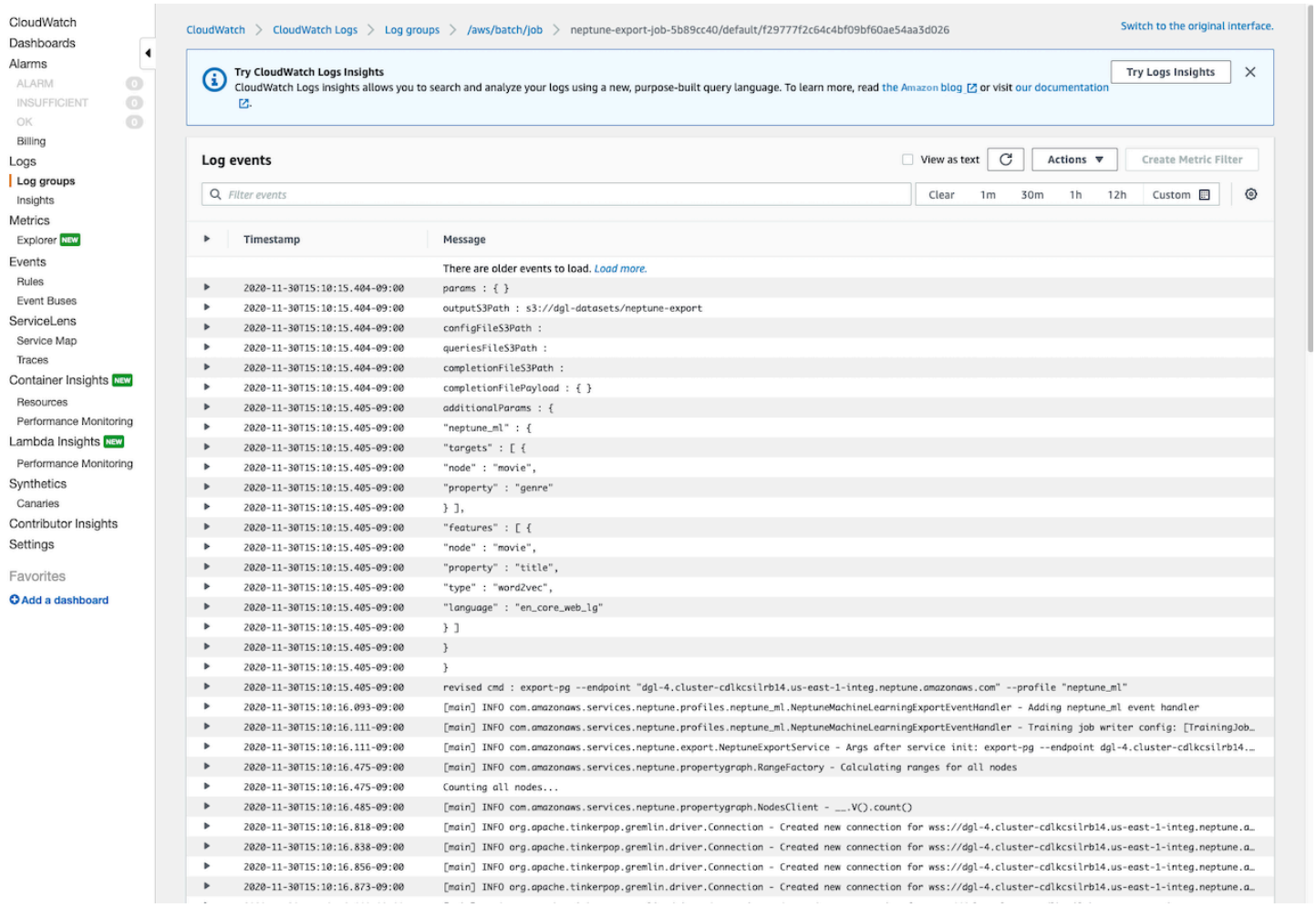
Se il servizio non ha ancora avviato il processo di esportazione, restituirà la risposta seguente:

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "pending"
}
```

Quando si ripete il comando dopo l'avvio del processo di esportazione, la risposta sarà simile alla seguente:

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "running",
  "logs": "https://us-east-1.console.aws.amazon.com/cloudwatch/home?..."
}
```

Se apri i log in File di log Amazon CloudWatch utilizzando l'URI fornito dalla chiamata di stato, puoi monitorare in dettaglio l'avanzamento dell'esportazione:



Annullamento di un processo di esportazione in esecuzione

Per annullare un processo di esportazione in esecuzione tramite la AWS Management Console

1. Apri la console AWS Batch all'indirizzo https://console.aws.amazon.com/batch/.
2. Scegliere Jobs (Processi).
3. Individua il processo in esecuzione da annullare, in base al `jobID` corrispondente.
4. Seleziona Annulla processo.

Per annullare un processo di esportazione in esecuzione tramite l'API Neptune-Export:

Invia una richiesta `HTTP DELETE` a `NeptuneExportApiUri` con l'aggiunta di `jobID`, in questo modo:

```
curl -X DELETE \
```

```
  (your NeptuneExportApiUri) (the job ID)
```

Utilizzo dello strumento da riga di comando **`neptune-export`** per esportare i dati da Neptune

Puoi utilizzare la procedura seguente per esportare dati dal tuo cluster database Neptune ad Amazon S3 con l'utilità da riga di comando `neptune-export`:

Prerequisiti per l'utilizzo dell'utilità da riga di comando **`neptune-export`**

Prima di iniziare

- Versione 8 di JDK: è necessario che sia installata la versione 8 di Java SE Development Kit (JDK).
- Scarica l'utilità neptune-export: scarica e installa il file neptune-export.jar.
- Assicurati che **`neptune-export`** abbia accesso al VPC Neptune: esegui neptune-export da una posizione in cui possa accedere al VPC in cui si trova il cluster database Neptune.

  Ad esempio, puoi eseguirlo su un'istanza Amazon EC2 all'interno del VPC Neptune o in un VPC separato collegato al VPC Neptune o su un host bastione separato.
- Assicurati che i gruppi di sicurezza VPC concedano l'accesso a **`neptune-export`**: verifica che i gruppi di sicurezza VPC collegati al VPC Neptune consentano l'accesso al tuo cluster database dall'indirizzo IP o dal gruppo di sicurezza associato all'ambiente `neptune-export`.
- Configura le autorizzazioni IAM necessarie: se nel database è abilitata l'autenticazione l'autenticazione AWS Identity and Access Management (IAM), assicurati che il ruolo in cui viene eseguita l'utilità `neptune-export` sia associato a una policy IAM che consenta le connessioni a Neptune. Per informazioni generali sulle policy Neptune, consulta Utilizzo delle policy IAM.

  Se desideri utilizzare il parametro di esportazione `clusterId` nelle richieste di query, il ruolo in cui viene eseguita l'utilità `neptune-export` richiede le seguenti autorizzazioni IAM:

  - `rds:DescribeDBClusters`
  - `rds:DescribeDBInstances`
  - `rds:ListTagsForResource`

  Se desideri esportare da un cluster clonato, il ruolo in cui viene eseguita l'utilità `neptune-export` richiede le seguenti autorizzazioni IAM:

  - `rds:AddTagsToResource`
  - `rds:DescribeDBClusters`
  - `rds:DescribeDBInstances`

- `rds:ListTagsForResource`
- `rds:DescribeDBClusterParameters`
- `rds:DescribeDBParameters`
- `rds:ModifyDBParameterGroup`
- `rds:ModifyDBClusterParameterGroup`
- `rds:RestoreDBClusterToPointInTime`
- `rds:DeleteDBInstance`
- `rds:DeleteDBClusterParameterGroup`
- `rds:DeleteDBParameterGroup`
- `rds:DeleteDBCluster`
- `rds:CreateDBInstance`
- `rds:CreateDBClusterParameterGroup`
- `rds:CreateDBParameterGroup`

Per pubblicare i dati esportati su Amazon S3, il ruolo in cui viene eseguita l'utilità `neptune-export` richiede le seguenti autorizzazioni IAM per le posizioni Amazon S3:

- `s3:PutObject`
- `s3:PutObjectTagging`
- `s3:GetObject`

- Imposta la variabile di ambiente **`SERVICE_REGION`**: imposta la variabile di ambiente `SERVICE_REGION` per identificare la regione in cui si trova il cluster database (consulta Connessione a Neptune per l'elenco degli identificatori di regione).

Esecuzione dell'utilità **`neptune-export`** per avviare un'operazione di esportazione

Usa il seguente comando per eseguire neptune-export dalla riga di comando e avviare un'operazione di esportazione:

```
java -jar neptune-export.jar nesvc \
  --root-path (path to a local directory) \
  --json (the JSON file that defines the export)
```

Il comando ha due parametri:

Parametri per neptune-export all'avvio di un'esportazione

- **`--root-path`**: percorso di una directory locale in cui i file di esportazione vengono scritti prima di essere pubblicati su Amazon S3.
- **`--json`**: oggetto JSON che definisce l'esportazione.

Comandi di esempio per l'utilità da riga di comando **`neptune-export`**

Per esportare i dati del grafo delle proprietà direttamente dal cluster database di origine:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
            }
          }'
```

Per esportare i dati RDF direttamente dal cluster database di origine:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-rdf",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
            }
          }'
```

Se si omette il parametro della richiesta `command`, per impostazione predefinita l'utilità `neptune-export` esporta i dati del grafo delle proprietà da Neptune.

Per esportare i dati da un clone del cluster database:

```
java -jar neptune-export.jar nesvc \
```

```
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)",
              "cloneCluster" : true
            }
          }'
```

Per esportare i dati dal cluster database utilizzando l'autenticazione IAM:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
              "useIamAuth" : true
            }
          }'
```

File esportati da Neptune-Export e **neptune-export**

Al termine dell'esportazione, i file di esportazione vengono pubblicati nella regione Amazon S3 specificata. Tutti i file pubblicati in Amazon S3 vengono crittografati tramite la crittografia lato server di Amazon S3 (`SSE-S3`). Le cartelle e i file pubblicati su Amazon S3 variano a seconda che si esportino dati del grafo delle proprietà o RDF. Se si apre la posizione di Amazon Amazon S3 in cui sono pubblicati i file, sarà visualizzato il seguente contenuto:

Posizioni dei file esportati in Amazon S3

- **`nodes/`**: questa cartella contiene i file di dati dei nodi in formato CSV (valori separati da virgole) o JSON.

  In Neptune i nodi possono avere una o più etichette. I nodi con etichette singole diverse (o combinazioni diverse di più etichette) vengono scritti in file differenti, affinché nessun singolo file contenga dati per nodi con combinazioni diverse di etichette. Se un nodo ha più etichette, queste vengono ordinate alfabeticamente prima di essere assegnate a un file.
- **`edges/`**: questa cartella contiene i file di dati degli archi in formato CSV (valori separati da virgole) o JSON.

  Come per i file dei nodi, i dati degli archi vengono scritti in file diversi in base a una combinazione delle relative etichette. Ai fini del training del modello, i dati degli archi vengono assegnati a file diversi in base a una combinazione dell'etichetta dell'arco e delle etichette del nodo iniziale e del nodo finale dell'arco.
- **`statements/`**: questa cartella contiene i file di dati RDF in formato Turtle, N-Quads, N-Triples o JSON.
- **`config.json`**: questo file contiene lo schema del grafo dedotto dal processo di esportazione.
- **`lastEventId.json`**: questo file contiene i valori `commitNum` e `opNum` dell'ultimo evento nei flussi Neptune del database. Il processo di esportazione include questo file solo se si imposta il parametro di esportazione `includeLastEventId` su `true` e se per il database da cui si esportano i dati sono abilitati i flussi Neptune.

Parametri utilizzati per controllare il processo di esportazione di Neptune

Sia che si utilizzi il servizio Neptune-Export o l'utilità da riga di comando `neptune-export`, i parametri per controllare l'esportazione sono per lo più gli stessi. Contengono un oggetto JSON passato all'endpoint Neptune-Export o a `neptune-export` nella riga di comando.

L'oggetto passato al processo di esportazione contiene fino a cinque campi di primo livello:

```
-d '{
      "command" : "(either export-pg or export-rdf)",
      "outputS3Path" : "s3:/(your Amazon S3 bucket)/(path to the folder for exported
 data)",
      "jobsize" : "(for Neptune-Export service only)",
      "params" : { (a JSON object that contains export-process parameters) },
      "additionalParams": { (a JSON object that contains parameters for training
 configuration) }
   }'
```

Indice

    - Combinazione di filter, gremlinNodeFilter, nodeLabels, edgeLabels e scope
- Filtro dell'esportazione dei dati RDF
    - Utilizzo di rdfExportScope e sparql per esportare archi specifici
    - Usato namedGraph per esportare un singolo grafico con nome

Parametro **`command`**

Il parametro `command` di primo livello determina se esportare i dati del grafo delle proprietà o i dati RDF. Se si omette il parametro `command`, per impostazione predefinita il processo di esportazione prevede l'esportazione dei dati del grafo delle proprietà.

- **`export-pg`**: esporta i dati del grafo delle proprietà.
- **`export-rdf`**: esporta i dati RDF.

Parametro **`outputS3Path`**

Il parametro `outputS3Path` di primo livello è obbligatorio e deve contenere l'URI di una posizione Amazon S3 in cui è possibile pubblicare i file esportati:

```
  "outputS3Path" : "s3://(your Amazon S3 bucket)/(path to output folder)"
```

Il valore deve iniziare con `s3://`, seguito da un nome di bucket valido e, facoltativamente, da un percorso di cartella all'interno del bucket.

Parametro **`jobSize`**

Il parametro `jobSize` di primo livello viene utilizzato solo con il servizio Neptune-Export, non con l'utilità da riga di comando `neptune-export`, ed è facoltativo. Consente di caratterizzare le dimensioni del processo di esportazione che si sta avviando, per determinare la quantità di risorse di calcolo dedicate al job e il livello massimo di concorrenza.

```
  "jobsize" : "(one of four size descriptors)"
```

I quattro descrittori di dimensione validi sono:

- `small`: concorrenza massima: 8. Adatto per volumi di archiviazione fino a 10 GB.

- `medium`: concorrenza massima: 32. Adatto per volumi di archiviazione fino a 100 GB.
- `large`: concorrenza massima: 64. Adatto per volumi di archiviazione superiori a 100 GB ma inferiori a 1 TB.
- `xlarge`: concorrenza massima: 96. Adatto per volumi di archiviazione superiori a 1 TB.

Per impostazione predefinita, un'esportazione avviata nel servizio Neptune-Export viene eseguita come processo `small`.

Le prestazioni di un'esportazione dipendono non solo dall'impostazione `jobSize`, ma anche dal numero di istanze database da cui si esegue l'esportazione, dalle dimensioni di ogni istanza e dal livello di concorrenza effettivo del processo.

Per le esportazioni del grafo delle proprietà, è possibile configurare il numero di istanze database utilizzando il parametro cloneClusterReplicaConta e configurare il livello di concorrenza effettivo del processo con il parametro simultaneità.

Oggetto **`params`**

Il parametro `params` di primo livello è un oggetto JSON che contiene i parametri utilizzati per controllare il processo di esportazione stesso, come illustrato in Campi dei parametri di esportazione nell'oggetto JSON `params` di primo livello. Alcuni campi nell'oggetto `params` sono specifici per le esportazioni dei grafi di proprietà, altri per i dati RDF.

Oggetto **`additionalParams`**

Il parametro `additionalParams` di primo livello è un oggetto JSON che contiene i parametri che è possibile utilizzare per controllare le azioni applicate ai dati dopo l'esportazione. Al momento, `additionalParams` viene utilizzato solo per esportare i dati di training per Neptune ML.

Campi dei parametri di esportazione nell'oggetto JSON **`params`** di primo livello

L'oggetto JSON `params` di esportazione Neptune consente di controllare l'esportazione, inclusi il tipo e il formato dei dati esportati.

Elenco dei campi possibili nell'oggetto **`params`** dei parametri di esportazione

Di seguito sono elencati tutti i possibili campi di primo livello che possono essere presenti in un oggetto `params`. In ogni oggetto può essere presente solo un sottoinsieme di questi campi.

Elenco dei campi comuni a tutti i tipi di esportazioni

- `cloneCluster`
- `cloneClusterInstanceType`
- `cloneClusterReplicaCount`
- `clusterId`
- `endpoint`
- `endpoints`
- `profile`
- `useIamAuth`
- `includeLastEventId`

Elenco dei campi per le esportazioni dei grafi di proprietà

- `concurrency`
- `edgeLabels`
- `filter`
- `filterConfigFile`
- `gremlinFilter`
- `gremlinNodeFilter`
- `gremlinEdgeFilter`
- `format`

- nodeLabels
- scope

Elenco dei campi per le esportazioni RDF

- format
- rdfExportScope
- sparql
- namedGraph

Campi comuni a tutti i tipi di esportazioni

Campo **`cloneCluster`** in **`params`**

(Facoltativo). Default: `false`.

Se il parametro `cloneCluster` è impostato su `true`, il processo di esportazione utilizza un clone rapido del cluster database:

```
  "cloneCluster" : true
```

Per impostazione predefinita, il processo di esportazione esporta i dati dal cluster database specificati utilizzando i parametri `endpoint`, `endpoints` o `clusterId`. Tuttavia, se il cluster database è in uso durante l'esportazione e i dati vengono modificati, il processo di esportazione non può garantire la coerenza dei dati esportati.

Per garantire la coerenza dei dati esportati, usa il parametro `cloneCluster` per esportare i dati da un clone statico del cluster database.

Il cluster database clonato viene creato nello stesso VPC del cluster database di origine ed eredita le impostazioni di autenticazione del gruppo di sicurezza, del gruppo di sottoreti e del database IAM dell'origine. Al termine dell'esportazione, Neptune elimina il cluster database clonato.

Per impostazione predefinita, un cluster database clonato è costituito da una singola istanza dello stesso tipo di istanza dell'istanza primaria nel cluster database di origine. È possibile modificare il tipo di istanza utilizzato per il cluster database clonato specificandone uno diverso utilizzando `cloneClusterInstanceType`.

> **Note**
>
> Se non si utilizza l'opzione `cloneCluster` e si esegue l'esportazione direttamente dal cluster database principale, potrebbe essere necessario aumentare il timeout nelle istanze da cui vengono esportati i dati. Per set di dati di grandi dimensioni, il timeout deve essere impostato su diverse ore.

Campo **`cloneClusterInstanceType`** in **`params`**

(Facoltativo).

Se il parametro `cloneCluster` è presente e impostato su `true`, è possibile usare il parametro `cloneClusterInstanceType` per specificare il tipo di istanza utilizzato per il cluster database clonato:

Per impostazione predefinita, un cluster database clonato è costituito da una singola istanza dello stesso tipo di istanza dell'istanza primaria nel cluster database di origine.

```
  "cloneClusterInstanceType" : "(for example, r5.12xlarge)"
```

Campo **`cloneClusterReplicaCount`** in **`params`**

(Facoltativo).

Se il parametro `cloneCluster` è presente e impostato su `true`, è possibile usare il parametro `cloneClusterReplicaCount` per specificare il numero di istanze di replica di lettura create nel cluster database clonato:

```
  "cloneClusterReplicaCount" : (for example, 3)
```

Per impostazione predefinita, un cluster database clonato è costituito da una singola istanza primaria. Il parametro `cloneClusterReplicaCount` consente di specificare il numero di istanze di replica di lettura aggiuntive da creare.

Campo **`clusterId`** in **`params`**

(Facoltativo).

Il parametro `clusterId` specifica l'ID del cluster database da usare:

```
"clusterId" : "(the ID of your DB cluster)"
```

Se si utilizza il parametro `clusterId`, il processo di esportazione usa tutte le istanze disponibili in tale cluster database per estrarre i dati.

> **Note**
>
> I parametri `endpoint`, `endpoints` e `clusterId` si escludono a vicenda. Devi usare solamente uno di essi.

Campo **`endpoint`** in **`params`**

(Facoltativo).

Usa `endpoint` per specificare un endpoint di un'istanza Neptune nel cluster database su cui il processo di esportazione può eseguire query per estrarre i dati (consulta Connessioni endpoint). Si tratta solo del nome DNS e non include il protocollo o la porta:

```
"endpoint" : "(a DNS endpoint of your DB cluster)"
```

Usa un cluster o un endpoint dell'istanza, ma non l'endpoint di lettura principale.

> **Note**
>
> I parametri `endpoint`, `endpoints` e `clusterId` si escludono a vicenda. Devi usare solamente uno di essi.

Campo **`endpoints`** in **`params`**

(Facoltativo).

Usa `endpoints` per specificare un array JSON di endpoint nel cluster database su cui il processo di esportazione può eseguire query per estrarre i dati (consulta Connessioni endpoint). Si tratta solo di nomi DNS e non includono il protocollo o la porta:

```
"endpoints": [
  "(one endpoint in your DB cluster)",
```

```
    "(another endpoint in your DB cluster)",
    "(a third endpoint in your DB cluster)"
    ]
```

Se nel cluster sono presenti più istanze (un'istanza primaria e una o più repliche di lettura), è possibile migliorare le prestazioni di esportazione utilizzando il parametro `endpoints` per distribuire le query su un elenco di tali endpoint.

> **Note**
>
> I parametri `endpoint`, `endpoints` e `clusterId` si escludono a vicenda. Devi usare solamente uno di essi.

Campo **`profile`** in **`params`**

(Obbligatorio per l'esportazione dei dati di training per Neptune ML, a meno che il campo *`neptune_ml`* non sia presente nel campo *`additionalParams`*).

Il parametro `profile` fornisce set di parametri preconfigurati per carichi di lavoro specifici. Al momento, il processo di esportazione supporta solo il profilo `neptune_ml`

Se devi esportare i dati di training per Neptune ML, aggiungi il seguente parametro all'oggetto `params`:

```
"profile" : "neptune_ml"
```

Campo **`useIamAuth`** in **`params`**

(Facoltativo). Default: `false`.

Se per il database da cui stai esportando i dati è abilitata l'autenticazione IAM, devi includere il parametro `useIamAuth` impostato su `true`:

```
"useIamAuth" : true
```

Campo **`includeLastEventId`** in **`params`**

Se imposti `includeLastEventId` su true e per il database da cui esporti i dati sono abilitati i flussi di Neptune, il processo di esportazione scrive un file `lastEventId.json` nella posizione di

esportazione specificata. Questo file contiene i valori `commitNum` e `opNum` dell'ultimo evento nel flusso.

```
  "includeLastEventId" : true
```

Un database clonato creato dal processo di esportazione eredita l'impostazione dei flussi del relativo database padre. Se i flussi sono abilitati per il database padre, saranno abilitati anche per il clone. Il contenuto del flusso sul clone rifletterà il contenuto del database padre (inclusi gli stessi ID di evento) al momento della creazione del clone.

Campi per l'esportazione dei grafi di proprietà

Campo **`concurrency`** in **`params`**

(Facoltativo). Default: `4`.

Il parametro `concurrency` specifica il numero di query parallele che il processo di esportazione deve utilizzare:

```
  "concurrency" : (for example, 24)
```

È consigliabile impostare il livello di concorrenza su un valore pari al doppio del numero di vCPU su tutte le istanze da cui si esportano i dati. Un'istanza r5.xlarge, ad esempio, ha 4 vCPU. Se si esegue l'esportazione da un cluster di 3 istanze r5.xlarge, è possibile impostare il livello di concorrenza su 24 (= 3 x 2 x 4).

Se si utilizza il servizio Neptune-Export, il livello di concorrenza è limitato dall'impostazione jobSize. Un processo di piccole dimensioni, ad esempio, supporta un livello di concorrenza pari a 8. Se si tenta di specificare un livello di concorrenza di 24 per un processo di piccole dimensioni utilizzando il parametro `concurrency`, il livello effettivo rimane impostato su 8.

Se si esportano i dati da un cluster clonato, il processo di esportazione calcola un livello di concorrenza appropriato in base alle dimensioni delle istanze clonate e alle dimensioni del processo.

Campo **`edgeLabels`** in **`params`**

(Facoltativo).

Usa `edgeLabels` per esportare solo gli archi con le etichette specificate:

```
  "edgeLabels" : ["(a label)", "(another label"]
```

Ogni etichetta nell'array JSON deve essere un'unica etichetta semplice.

Il parametro `scope` ha la precedenza sul parametro `edgeLabels`, quindi se il valore `scope` non include archi, il parametro `edgeLabels` non ha alcun effetto.

Campo **`filter`** in **`params`**

(Facoltativo).

Usa `filter` per specificare che devono essere esportati solo i nodi e/o gli archi con etichette specifiche e per filtrare le proprietà esportate per ogni nodo o arco.

La struttura generale di un oggetto `filter`, in linea o in un file di configurazione del filtro, è la seguente:

```
  "filter" : {
    "nodes": [ (array of node label and properties objects) ],
    "edges": [ (array of edge definition an properties objects) ]
  }
```

- **`nodes`**: contiene un array JSON di nodi e proprietà dei nodi nel seguente formato:

  ```
    "nodes : [
      {
        "label": "(node label)",
        "properties": [ "(a property name)", "(another property name)", ( ... ) ]
      }
    ]
  ```

  - `label`: l'etichetta o le etichette del grafo delle proprietà del nodo.

    Accetta un singolo valore o, se il nodo contiene più etichette, un array di valori.
  - `properties`: contiene un array dei nomi delle proprietà del nodo che si desidera esportare.
- **`edges`**: contiene un array JSON di definizioni degli archi nel seguente formato:

  ```
    "edges" : [
      {
        "label": "(edge label)",
  ```

```
        "properties": [ "(a property name)", "(another property name)", ( ... ) ]
      }
    ]
```

- `label`: etichetta del grafo delle proprietà dell'arco. Accetta un valore singolo.
- `properties`: contiene un array dei nomi delle proprietà dell'arco che si desidera esportare.

Campo **`filterConfigFile`** in **`params`**

(Facoltativo).

Usa `filterConfigFile` per specificare un file JSON che contiene una configurazione del filtro nello stesso formato accettato dal parametro `filter`:

```
 "filterConfigFile" : "s3://(your Amazon S3 bucket)/neptune-export/(the name of the
JSON file)"
```

Consulta filter per il formato del file `filterConfigFile`.

Campo **`format`** usato per i dati del grafo delle proprietà in **`params`**

(Facoltativo). Valore predefinito: `csv` (valori separati da virgole)

Il parametro `format` specifica il formato di output dei dati del grafo delle proprietà esportati:

```
 "format" : (one of: csv, csvNoHeaders, json, neptuneStreamsJson)
```

- **`csv`**: output in formato CSV (valori separati da virgole), con intestazioni di colonna formattate in base al formato dei dati di caricamento Gremlin.
- **`csvNoHeaders`**: dati in formato CSV senza intestazioni di colonna.
- **`json`**: dati in formato JSON.
- **`neptuneStreamsJson`**: dati in formato JSON che utilizzano il formato di serializzazione delle modifiche GREMLIN_JSON.

Campo **`gremlinFilter`** in **`params`**

(Facoltativo).

Il parametro `gremlinFilter` permette di specificare un frammento Gremlin, ad esempio una fase `has()`, che consente di filtrare sia i nodi che gli archi:

```
"gremlinFilter" : (a Gremlin snippet)
```

I nomi dei campi e i valori delle stringhe devono essere racchiusi tra virgolette doppie precedute da caratteri di escape. Per date e ore, puoi usare il metodo datetime.

L'esempio seguente esporta solo i nodi e gli archi con una proprietà date-created il cui valore è maggiore di 2021-10-10:

```
"gremlinFilter" : "has(\"created\", gt(datetime(\"2021-10-10\")))"
```

Campo **`gremlinNodeFilter`** in **`params`**

(Facoltativo).

Il parametro `gremlinNodeFilter` permette di specificare un frammento Gremlin, ad esempio una fase `has()`, che consente di filtrare i nodi:

```
"gremlinNodeFilter" : (a Gremlin snippet)
```

I nomi dei campi e i valori delle stringhe devono essere racchiusi tra virgolette doppie precedute da caratteri di escape. Per date e ore, puoi usare il metodo datetime.

L'esempio seguente esporta solo i nodi con una proprietà `deleted` booleana il cui valore è `true`:

```
"gremlinNodeFilter" : "has(\"deleted\", true)"
```

Campo **`gremlinEdgeFilter`** in **`params`**

(Facoltativo).

Il parametro `gremlinEdgeFilter` permette di specificare un frammento Gremlin, ad esempio una fase `has()`, che consente di filtrare gli archi:

```
"gremlinEdgeFilter" : (a Gremlin snippet)
```

I nomi dei campi e i valori delle stringhe devono essere racchiusi tra virgolette doppie precedute da caratteri di escape. Per date e ore, puoi usare il metodo datetime.

L'esempio seguente esporta solo gli archi con una proprietà `strength` numerica il cui valore è 5:

```
  "gremlinEdgeFilter" : "has(\"strength\", 5)"
```

Campo **`nodeLabels`** in **`params`**

(Facoltativo).

Usa `nodeLabels` per esportare solo i nodi con le etichette specificate:

```
  "nodeLabels" : ["(a label)", "(another label"]
```

Ogni etichetta nell'array JSON deve essere un'unica etichetta semplice.

Il parametro `scope` ha la precedenza sul parametro `nodeLabels`, quindi se il valore `scope` non include nodi, il parametro `nodeLabels` non ha alcun effetto.

Campo **`scope`** in **`params`**

(Facoltativo). Default: `all`.

Il parametro `scope` specifica se esportare solo nodi, solo archi o entrambi:

```
  "scope" : (one of: nodes, edges, or all)
```

- `nodes`: esporta solo i nodi e le relative proprietà.
- `edges`: esporta solo gli archi e le relative proprietà.
- `all`: esporta sia i nodi che gli archi e le relative proprietà (impostazione predefinita).

Campi per l'esportazione dei dati RDF

Campo **`format`** utilizzato per i dati RDF in **`params`**

(Facoltativo). Default: `turtle`

Il parametro `format` specifica il formato di output dei dati RDF esportati:

```
  "format" : (one of: turtle, nquads, ntriples, neptuneStreamsJson)
```

- **`turtle`**: output in formato Turtle.
- **`nquads`**: dati in formato N-Quads senza intestazioni di colonna.
- **`ntriples`**: dati in formato N-Triples.
- **`neptuneStreamsJson`**: dati in formato JSON che utilizzano il formato di serializzazione delle modifiche SPARQL NQUADS.

Campo **`rdfExportScope`** in **`params`**

(Facoltativo). Default: `graph`.

Il parametro `rdfExportScope` specifica l'ambito dell'esportazione RDF:

```
  "rdfExportScope" : (one of: graph, edges, or query)
```

- `graph`: esporta tutti i dati RDF.
- `edges`: esporta solo le triple che rappresentano gli archi.
- `query`: esporta i dati recuperati da una query SPARQL fornita utilizzando il campo `sparql`.

Campo **`sparql`** in **`params`**

(Facoltativo).

Il parametro `sparql` consente di specificare una query SPARQL per recuperare i dati da esportare:

```
  "sparql" : (a SPARQL query)
```

Se si fornisce una query utilizzando il campo `sparql`, è anche necessario impostare il campo `rdfExportScope` su `query`.

Campo **`namedGraph`** in **`params`**

(Facoltativo).

Il `namedGraph` parametro consente di specificare un IRI per limitare l'esportazione a un singolo grafico denominato:

```
  "namedGraph" : (Named graph IRI)
```

Il `namedGraph` parametro può essere utilizzato solo con il `rdfExportScope` campo impostato su`graph`.

Esempi di filtro dei dati esportati

Di seguito sono riportati alcuni esempi che illustrano i modi per filtrare i dati esportati.

Filtro dell'esportazione dei dati del grafo delle proprietà

Esempio di utilizzo di **`scope`** per esportare solo gli archi

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "scope": "edges"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Esempio di utilizzo di **`nodeLabels`** e **`edgeLabels`** per esportare solo nodi e archi con etichette specifiche

Il parametro `nodeLabels` nell'esempio seguente specifica che devono essere esportati solo i nodi con un'etichetta `Person` o `Post`. Il parametro `edgeLabels` specifica che devono essere esportati solo gli archi con un'etichetta `likes`:

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "nodeLabels": ["Person", "Post"],
    "edgeLabels": ["likes"]
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Esempio di utilizzo di **`filter`** per esportare solo i nodi, gli archi e le proprietà specificati

L'oggetto `filter` in questo esempio esporta i nodi `country` i nodi con le relative proprietà `type`, `code` e `desc` e anche gli archi `route` con la relativa proprietà `dist`.

```
{
  "command": "export-pg",
  "params": {
```

```
    "endpoint": "(your Neptune endpoint DNS name)",
    "filter": {
      "nodes": [
        {
          "label": "country",
          "properties": [
            "type",
            "code",
            "desc"
          ]
        }
      ],
      "edges": [
        {
          "label": "route",
          "properties": [
            "dist"
          ]
        }
      ]
    }
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Esempio che utilizza **`gremlinFilter`**.

Questo esempio utilizza `gremlinFilter` per esportare solo i nodi e gli archi creati dopo il 2021-10-10 (ovvero, con una proprietà `created` il cui valore è maggiore del 2021-10-10):

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinFilter" : "has(\"created\", gt(datetime(\"2021-10-10\")))"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Esempio che utilizza **`gremlinNodeFilter`**.

Questo esempio utilizza `gremlinNodeFilter` per esportare solo i nodi eliminati (nodi con una proprietà `deleted` booleana il cui valore è `true`):

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinNodeFilter" : "has(\"deleted\", true)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Esempio che utilizza **`gremlinEdgeFilter`** .

Questo esempio usa `gremlinEdgeFilter` per esportare solo gli archi con una proprietà `strength` numerica il cui valore è 5:

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinEdgeFilter" : "has(\"strength\", 5)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Combinazione di **`filter`**, **`gremlinNodeFilter`**, **`nodeLabels`**, **`edgeLabels`** e **`scope`**

L'oggetto `filter` in questo esempio esporta:

- nodi `country` con le relative proprietà `type`, `code` e `desc`
- nodi `airport` con le relative proprietà `code`, `icao` e `runways`
- archi `route` con la relativa proprietà `dist`

Il parametro `gremlinNodeFilter` filtra i nodi in modo che vengano esportati solo i nodi con una proprietà `code` il cui valore inizia con A.

I parametri `nodeLabels` e `edgeLabels` limitano ulteriormente l'output in modo che vengano esportati solo i nodi `airport` e gli archi `route`.

Infine, il parametro `scope` elimina gli spigoli dall'esportazione, lasciando nell'output solo i nodi `airport` designati.

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "filter": {
      "nodes": [
        {
          "label": "airport",
          "properties": [
            "code",
            "icao",
            "runways"
          ]
        },
        {
          "label": "country",
          "properties": [
            "type",
            "code",
            "desc"
          ]
        }
      ],
      "edges": [
        {
          "label": "route",
          "properties": [
            "dist"
          ]
        }
      ]
    },
    "gremlinNodeFilter": "has(\"code\", startingWith(\"A\"))",
    "nodeLabels": [
      "airport"
    ],
    "edgeLabels": [
      "route"
    ],
    "scope": "nodes"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Filtro dell'esportazione dei dati RDF

Utilizzo di **`rdfExportScope`** e **`sparql`** per esportare archi specifici

Questo esempio esporta le triple il cui predicato è <http://kelvinlawrence.net/air-routes/objectProperty/route> e il cui oggetto non è un valore letterale:

```
{
  "command": "export-rdf",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "rdfExportScope": "query",
    "sparql": "CONSTRUCT { ?s <http://kelvinlawrence.net/air-routes/objectProperty/
route> ?o } WHERE { ?s ?p ?o . FILTER(!isLiteral(?o)) }"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Usato **`namedGraph`** per esportare un singolo grafico con nome

Questo esempio esporta le triple appartenenti al grafico denominato < http://aws.amazon.com/neptune/vocab/v01/ >: DefaultNamedGraph

```
{
  "command": "export-rdf",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "rdfExportScope": "graph",
    "namedGraph": "http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Risoluzione dei problemi relativi al processo di esportazione di Neptune

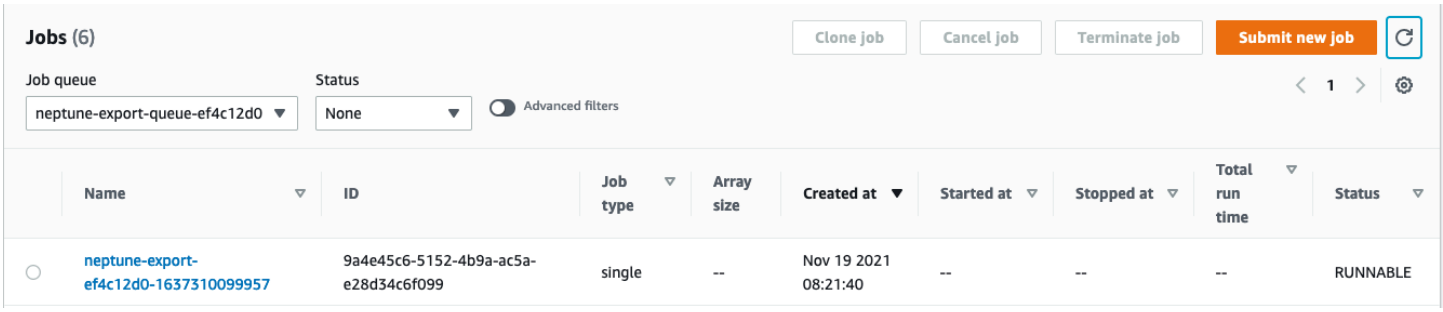
Il processo di esportazione di Amazon Neptune usa AWS Batch per effettuare il provisioning delle risorse di calcolo e archiviazione necessarie per esportare i dati di Neptune. Quando l'esportazione è in esecuzione, puoi utilizzare il link nel campo `logs` per accedere ai log di CloudWatch per il processo di esportazione.

Tuttavia, i log di CloudWatch per il processo AWS Batch che esegue l'esportazione sono disponibili solo quando il processo AWS Batch è in esecuzione. Se Neptune-Export segnala che un'esportazione è in sospeso, non sarà presente un link ai log tramite cui accedere ai log di CloudWatch. Se un processo di esportazione rimane nello stato `pending` per più di qualche minuto, potrebbe essersi verificato un problema durante il provisioning delle risorse AWS Batch sottostanti.

Quando il processo di esportazione esce dallo stato in sospeso, puoi verificarne lo stato nel modo seguente:

Per verificare lo stato di un processo AWS Batch

1. Aprire la console AWS Batch all'indirizzo https://console.aws.amazon.com/batch/.
2. Seleziona la coda dei processi neptune-export.
3. Cerca il processo con il nome corrispondente al valore di `jobName` restituito da Neptune quando hai avviato l'esportazione.

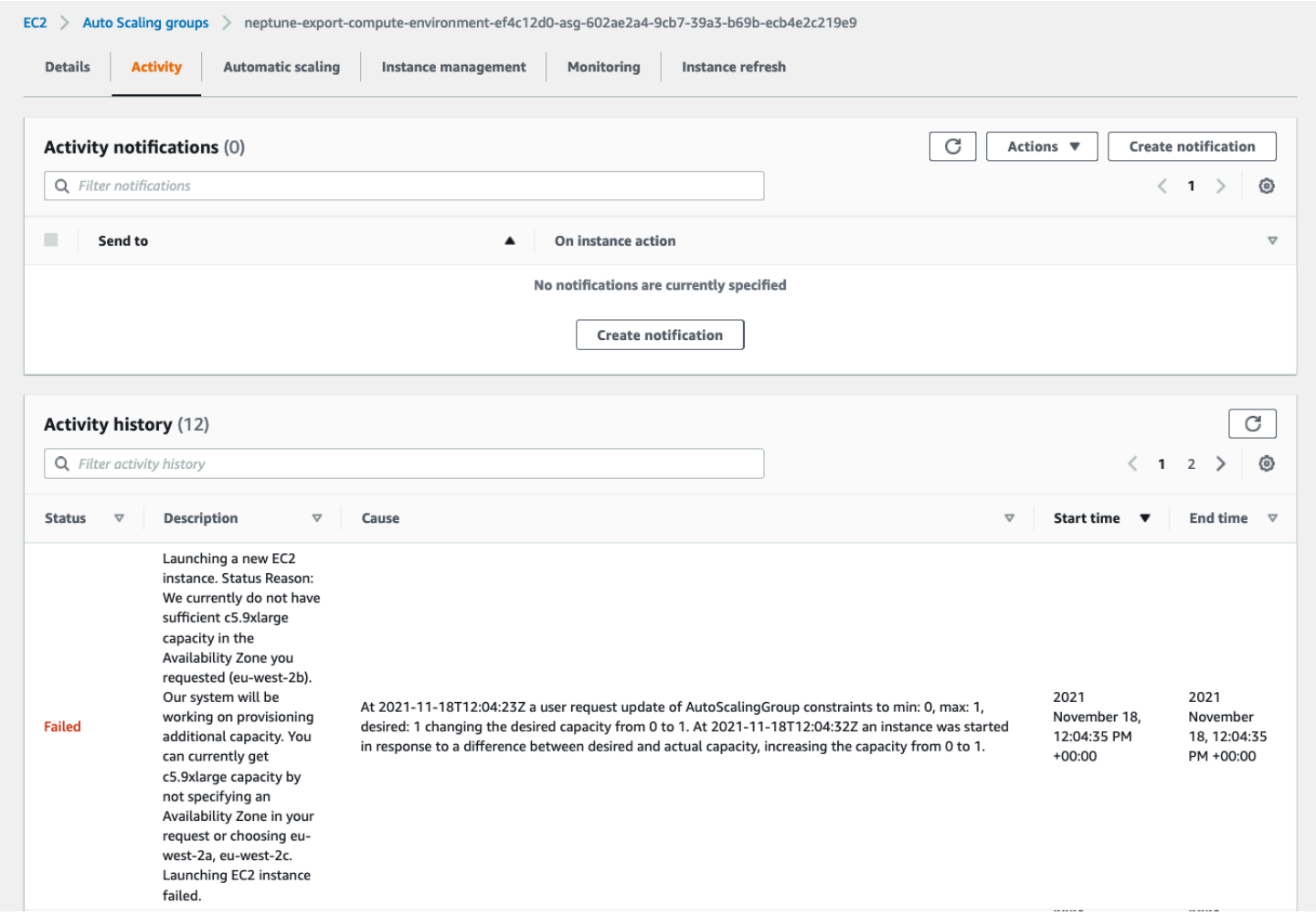


Se il processo rimane bloccato in uno stato `RUNNABLE`, è possibile che problemi di rete o di sicurezza impediscano all'istanza di container di essere aggiunta al cluster Amazon Elastic Container Service (Amazon ECS) sottostante. Consulta la sezione sulla verifica delle impostazioni di rete e di sicurezza dell'ambiente di calcolo in questo articolo di supporto.

Puoi anche verificare la presenza di problemi di dimensionamento automatico:

Per verificare il gruppo di dimensionamento automatico Amazon EC2 per l'ambiente di calcolo AWS Batch

1. Apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Seleziona il gruppo con dimensionamento automatico per l'ambiente di calcolo neptune-export.
3. Apri la scheda Attività e controlla la cronologia delle attività per verificare la presenza di eventi non riusciti.



Errori comuni di Neptune-Export

`org.eclipse.rdf4j.query.QueryEvaluationException: Tag mismatch!`

Se un processo `export-rdf` ha regolarmente esito negativo con un errore `Tag mismatch! QueryEvaluationException`, l'istanza Neptune è sottodimensionata per le query di grandi dimensioni e di lunga durata utilizzate da Neptune-Export.

È possibile evitare questo errore passando a un'istanza Neptune di dimensioni superiori o configurando il processo per l'esportazione da un cluster clonato di grandi dimensioni, in questo modo:

```
'{
  "command": "export-rdf",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "cloneCluster": True,
    "cloneClusterInstanceType" : "r5.24xlarge"
  }
}'
```

Gestione del database Amazon Neptune

In questa sezione viene descritto come gestire e mantenere il cluster database Neptune mediante la AWS Management Console e AWS CLI.

Neptune opera su cluster di server di database che sono collegati in una topologia di replica. Di conseguenza, la gestione di Neptune spesso implica l'implementazione di modifiche in più server e la necessità di accertarsi che tutte le repliche Neptune siano in grado di stare al passo con il server principale.

Poiché Neptune dimensiona in modo trasparente lo storage sottostante in base alla crescita dei dati, la gestione di Neptune richiede relativamente poca gestione dello storage su disco. Analogamente, dato che Neptune esegue backup continui in automatico, un cluster Neptune non richiede pianificazione o tempi di inattività per l'esecuzione dei backup.

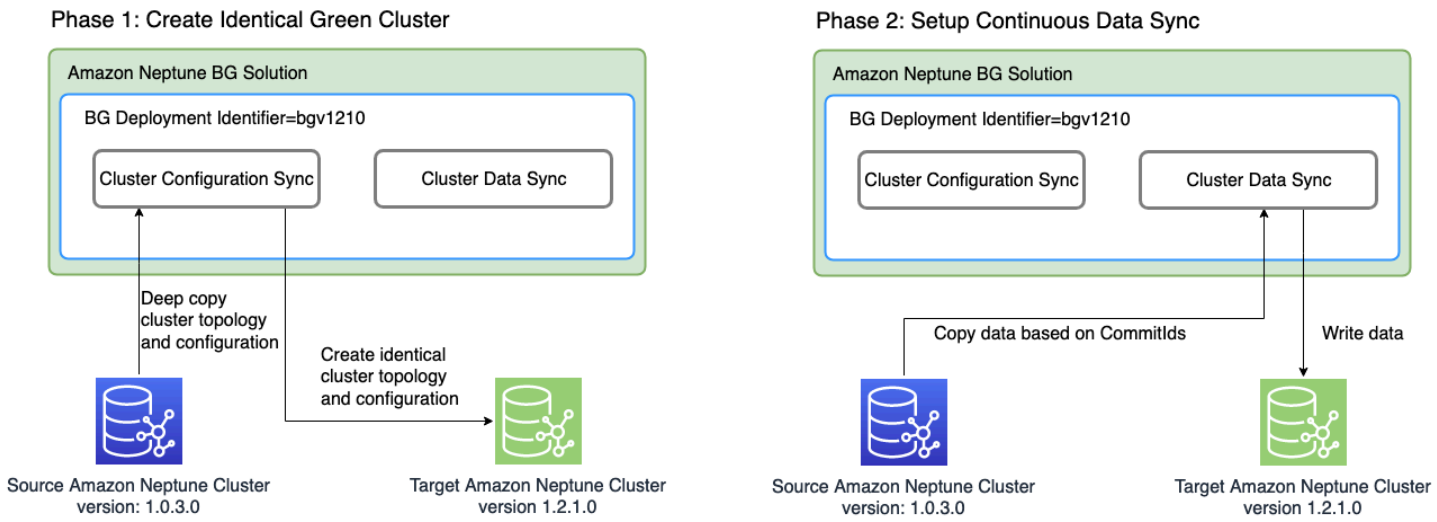
Argomenti

Utilizzo della soluzione blu/verde Neptune per l'esecuzione degli aggiornamenti blu/verde

Gli aggiornamenti del motore Amazon Neptune possono prevedere tempi di inattività delle applicazioni perché il database non è disponibile durante l'installazione e la verifica degli aggiornamenti. Questo vale indipendentemente dal fatto che vengano avviati manualmente o automaticamente.

Neptune offre una soluzione di implementazione blu/verde che è possibile eseguire usando uno stack AWS CloudFormation che riduce in modo significativo tali tempi di inattività. Crea un ambiente di gestione temporanea verde sincronizzato con l'ambiente di produzione blu. È quindi possibile aggiornare l'ambiente di gestione temporanea per eseguire un aggiornamento della versione secondaria o principale del motore, una modifica del modello dei dati del grafo o un aggiornamento del sistema operativo e testare il risultato. Infine, puoi effettuare rapidamente lo switchover all'ambiente di produzione, con tempi di inattività minimi.

La soluzione blu/verde Neptune prevede due fasi, come illustrato in questo diagramma:



La fase 1 crea un cluster database verde identico al cluster di produzione

La soluzione crea un cluster database con un identificatore di implementazione blu/verde univoco e con la stessa topologia di cluster del cluster di produzione. Ovvero ha lo stesso numero e dimensioni di istanze database, gli stessi gruppi di parametri e tutte le stesse configurazioni del cluster database di produzione (blu), tranne per il fatto che è stato aggiornato alla versione del motore di destinazione specificata, che deve essere successiva alla versione del motore corrente (blu). È possibile specificare una versione secondaria e una versione principale del motore per la destinazione.

Se necessario, la soluzione eseguirà tutti gli aggiornamenti intermedi al fine di raggiungere la versione del motore di destinazione specificata. Questo nuovo cluster diventa l'ambiente di gestione temporanea verde.

La fase 2 configura la sincronizzazione continua dei dati

Dopo che l'ambiente verde è stato completamente preparato, la soluzione imposta la replica continua tra il cluster di origine (blu) e il cluster di destinazione (verde) utilizzando i flussi Neptune. Quando la differenza di replica tra di essi raggiunge lo zero, l'ambiente di gestione temporanea è pronto per il test. A quel punto è necessario sospendere la scrittura sul cluster blu per evitare ulteriori ritardi di replica.

La versione del motore di destinazione potrebbe includere nuove funzionalità o dipendenze che influiscono sulle applicazioni. Controlla la pagina di rilascio del motore di destinazione e le pagine dei rilasci del motore intermedi nella sezione Rilasci del motore per informazioni su cosa è cambiato rispetto alla versione corrente del motore. È consigliabile eseguire test di integrazione o verificare manualmente le applicazioni nel cluster verde prima di promuoverle all'ambiente di produzione.

Dopo aver testato e valutato le modifiche nel cluster verde, è sufficiente cambiare l'endpoint del database nelle applicazioni dal cluster blu a quello verde.

Dopo lo switchover, la soluzione blu/verde Neptune non elimina l'ambiente di produzione blu precedente. Potrai continuare ad accedervi per ulteriori convalide e test, se necessario. Vengono addebitate le spese di fatturazione standard per le relative istanze fino a quando non le elimini. La soluzione blu/verde utilizza anche altri servizi AWS, i cui costi vengono fatturati ai prezzi normali. Le informazioni relative all'eliminazione della soluzione quando non è più necessaria sono riportate nella sezione Pulizia

Prerequisiti per l'esecuzione dello stack blu/verde Neptune

Prima di avviare lo stack blu/verde Neptune:

- Assicurati di abilitare i flussi Neptune nel cluster di produzione (blu).
- Tutte le istanze del cluster blu devono essere nello stato disponibile. Puoi controllare gli stati delle istanze nella console Neptune o usando l'API describe-db-instances.
- Tutte le istanze devono inoltre essere sincronizzate con il gruppo di parametri del cluster database.
- La soluzione blu/verde Neptune richiede un endpoint VPC DynamoDB nel VPC in cui si trova il cluster blu. Consulta Utilizzo di endpoint Amazon VPC per accedere a DynamoDB.

- Esegui la soluzione quando il carico di lavoro di scrittura sul cluster database di produzione blu è il più leggero possibile. Evita, ad esempio, di eseguire la soluzione nel corso di un caricamento in blocco o quando è probabile che si verifichi un numero elevato di operazioni di scrittura per qualsiasi altro motivo.

Utilizzo di un modello AWS CloudFormation per l'esecuzione della soluzione blu/verde Neptune

È possibile usare AWS CloudFormation per implementare la soluzione blu/verde Neptune. Il modello CloudFormation crea un'istanza Amazon EC2 nello stesso VPC del database Neptune di origine blu, installa la soluzione in quella posizione e la esegue. Puoi monitorarne l'avanzamento nei file di log CloudWatch, come spiegato in Monitoraggio dell'avanzamento.

Puoi usare questi collegamenti per verificare il modello di soluzione o selezionare il pulsante Avvia lo stack per avviarlo nella console AWS CloudFormation:

Visualizzazione	Visualizzazione in Designer	

Nella console scegli la regione AWS in cui desideri eseguire la soluzione dal menu a discesa nella parte superiore destra della finestra.

Imposta i parametri dello stack come segue:

- **`DeploymentID`**: identificatore univoco per ogni implementazione blu/verde Neptune.

  Viene utilizzato come identificatore del cluster database verde e come prefisso per assegnare i nomi alle nuove risorse create durante l'implementazione.
- **`NeptuneSourceClusterId`**: identificatore del cluster database blu da aggiornare.
- **`NeptuneTargetClusterVersion:`**: versione del motore Neptune a cui si desidera aggiornare il cluster database blu.

  Questo valore deve essere maggiore rispetto a quello della versione del motore corrente del cluster database blu.
- **`DeploymentMode`**: indica se si tratta di una nuova implementazione o di un tentativo di riprendere un'implementazione precedente. Se utilizzi lo stesso `DeploymentID` di un'implementazione precedente, imposta `DeploymentMode` su `resume`.

I valori validi sono `new` (predefinito) e `resume`.

- **`GraphQueryType`**: tipo di dati del grafo per il database.

  I valori validi sono `propertygraph` (predefinito) e `rdf`.

- **`SubnetId`**: ID di sottorete dello stesso VPC in cui si trova il cluster database blu. Consulta Connessione a un cluster database Neptune da un'istanza Amazon EC2 nello stesso VPC.

  Specifica l'ID di una sottorete pubblica per stabilire una connessione SSH all'istanza tramite EC2 Connect.

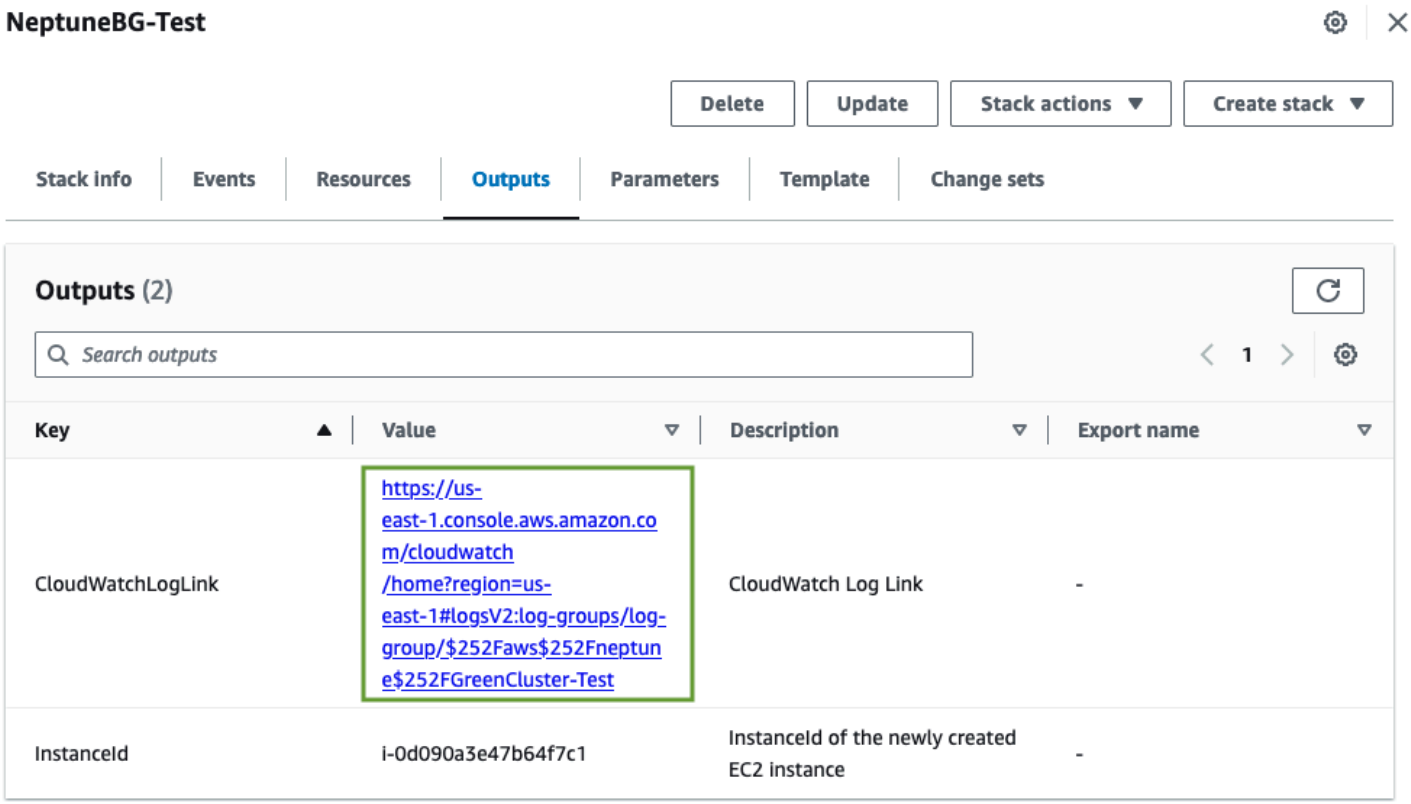
- **`InstanceSecurityGroup`**: gruppo di sicurezza per l'istanza Amazon EC2.

  Il gruppo di sicurezza deve avere accesso al cluster database blu e devi poter accedere all'istanza tramite SSH. Per informazioni, consultare Creare un gruppo di sicurezza tramite la console VPC.

Attendi che lo stack sia completo. Al termine, la soluzione verrà avviata. Potrai quindi monitorare il processo di implementazione utilizzando i file di log CloudWatch come descritto nella sezione successiva.

Monitoraggio dell'avanzamento dell'implementazione blu/verde Neptune

Per monitorare lo stato di avanzamento della soluzione blu/verde Neptune, accedi alla console CloudWatch e osserva i log nel gruppo di file di log CloudWatch per `/aws/neptune/`*`(Neptune Blue/Green deployment ID)`*. Puoi trovare un collegamento ai file di log CloudWatch negli output dello stack AWS CloudFormation della soluzione:



Se hai specificato una sottorete pubblica come parametro dello stack, puoi anche accedere tramite SSH all'istanza Amazon EC2 creata come parte dello stack e fare riferimento al file di log in `/var/log/cloud-init-output.log`.

Il file di log mostra le azioni intraprese dalla soluzione blu/verde Neptune, come mostrato in questo screenshot:

```
============================================================

Neptune Blue Green Deployment Solution Version: 0.1.06012023

============================================================

Checking whether cluster with id = bg-06-01-14-20-29test-bg1-bgInt already exists.

BlueGreen deployment_mode = new

Didn't find any cluster with id bg-06-01-14-20-29test-bg1-bgInt

Cloned_cluster_id: bg-06-01-14-20-29test-bg1-bgInt

Replication_stack_name: bg-06-01-14-20-29test-bg1-bgInt-replication

DescribeDbClusters response for test-bg1-bgIntegTest-06-01-14-20-29: {'AllocatedStorage': 1,
'AvailabilityZones': ['us-east-1b', 'us-east-1c', 'us-east-1f'], 'BackupRetentionPeriod': 1,
'DBClusterIdentifier': 'test-bg1-bgintegtest-06-01-14-20-29', 'DBClusterParameterGroup': 'green-     -blue-
green-deployment-test-123456789012345-pg-tes710', 'DBSubnetGroup': 'default', 'Status': 'available',
'EarliestRestorableTime': datetime.datetime(2023, 6, 1, 8, 51, 23, 394000, tzinfo=tzlocal()), 'Endpoint':
'test-bg1-bgintegtest-06-01-14-20-29.cluster-critvszpmydm.us-east-1.neptune.amazonaws.com', 'ReaderEndpoint':
'test-bg1-bgintegtest-06-01-14-20-29.cluster-ro-critvszpmydm.us-east-1.neptune.amazonaws.com', 'MultiAZ':
False, 'Engine': 'neptune', 'EngineVersion': '1.2.0.0', 'LatestRestorableTime': datetime.datetime(2023, 6, 1,
8, 51, 23, 394000, tzinfo=tzlocal()), 'Port': 8182, 'MasterUsername': 'admin', 'PreferredBackupWindow':
'06:33-07:03', 'PreferredMaintenanceWindow': 'fri:09:44-fri:10:14', 'ReadReplicaIdentifiers': [],
'DBClusterMembers': [{'DBInstanceIdentifier': 'test-bg1-bgintegtest-06-01-14-20-29i-1', 'IsClusterWriter':
True, 'DBClusterParameterGroupStatus': 'in-sync', 'PromotionTier': 1}], 'VpcSecurityGroups':
```

I messaggi del file di log mostrano lo stato di sincronizzazione tra i cluster blu e verde:

```
DDB checkpoint {'S': '1'}, {'S': '6'}

DDB Checkpoint: {'checkpointSubSequenceNumber': {'S': '6'}, 'lastUpdateTime': {'N': '1685611142127'},
'leaseOwner': {'S': 'nobody'}, 'checkpoint': {'S': '1'}, 'leaseKey': {'S': 'bg-anl   -234567899-replication'}}

Time difference for last checkpoint and last stream event: 5841351

Stream eventId difference for last replication checkpoint and last stream event on the Source cluster: 0:0

Found region : us-east-1

Cloudwatch Log Url for blue green solution is https://us-east-1.console.aws.amazon.com/cloudwatch
/home?region=us-east-1#logsV2:log-groups/log-group/aws/neptune/bg

Cloudwatch dashboard url for replication is https://console.aws.amazon.com/cloudwatch/home?region=us-
east-1#dashboards:name=neptune-stream-poller-bg-an   -234567899-replication

Replication poller lambda arn is arn:aws:lambda:us-east-1:451235071234:function:bg-an   -234567899-replic-
NeptuneStreamPollerLambd-B6V1ytULgmSP. Look for CW log the poller lambda for more troubleshooting.

Stream Last EventId {'commitNum': 1, 'opNum': 6} on cluster : database-d61852469-t        -experiment.cluster-
critvszpmydm.us-east-1.neptune.amazonaws.com:8182

DDB checkpoint {'S': '1'}, {'S': '6'}

DDB Checkpoint: {'checkpointSubSequenceNumber': {'S': '6'}, 'lastUpdateTime': {'N': '1685611207245'},
'leaseOwner': {'S': 'nobody'}, 'checkpoint': {'S': '1'}, 'leaseKey': {'S': 'bg-ankig-234567899-replication'}}
```

Il processo di sincronizzazione verifica il ritardo di replica calcolando la differenza tra il flusso `eventID` più recente nel cluster blu e il checkpoint di replica presente nella tabella dei checkpoint DynamoDB creata dallo stack di replica Neptune-Neptune. Utilizzando questi messaggi, è possibile monitorare la differenza di replica corrente.

Passaggio dal cluster blu di produzione al cluster verde aggiornato

Prima di promuovere il cluster verde all'ambiente di produzione, assicurati che la differenza di commit tra il cluster blu e il cluster verde sia pari a zero, quindi disabilita tutto il traffico di scrittura sul cluster blu. Se si continua a scrivere sul cluster blu durante il passaggio dell'endpoint del database al cluster verde potrebbe verificarsi un danneggiamento dei dati causato dalla scrittura di dati parziali su entrambi i cluster. Potrebbe non essere ancora il momento di disabilitare il traffico di lettura.

Se hai abilitato l'autenticazione IAM nel cluster di origine (blu), assicurati di aggiornare tutte le policy IAM utilizzate nelle tue applicazioni affinché puntino al cluster verde (per un esempio di tale policy, consulta policy di accesso senza restrizioni).

Dopo aver disabilitato il traffico di scrittura, attendi che termini la replica, quindi abilita il traffico di scrittura sul cluster verde (ma non sul cluster blu). Passa anche il traffico di lettura dal cluster blu a quello verde.

Pulizia dopo il completamento della soluzione blu/verde Neptune

Dopo aver promosso il cluster di gestione temporanea (verde) all'ambiente di produzione, effettua la pulizia delle risorse create dalla soluzione blu/verde Neptune:

- Elimina l'istanza Amazon EC2 creata per eseguire la soluzione.
- Elimina i modelli AWS CloudFormation per la replica basata su flussi Neptune che ha mantenuto il cluster verde sincronizzato con il cluster blu. Quello principale ha il nome dello stack che hai specificato in precedenza e uno è costituito dall'ID di implementazione seguito da "-replication", ovvero *(DeploymentID)*`-replication`.

L'eliminazione dei modelli AWS CloudFormation non elimina i cluster stessi. Dopo aver verificato che il cluster verde funzioni come previsto, puoi facoltativamente acquisire uno snapshot prima di eliminare manualmente il cluster blu.

Best practice per la soluzione blu/verde Neptune

- Prima di procedere con lo switchover del cluster verde all'ambiente di produzione, è consigliabile verificarne accuratamente il corretto funzionamento. Verifica la coerenza dei dati e la configurazione del database. È possibile che alcune delle nuove versioni del motore richiedano anche aggiornamenti del client. Controlla le note di rilascio della versione del motore prima di effettuare l'aggiornamento. È opportuno testare tutti questi aspetti negli ambienti di sviluppo, test e pre-produzione prima di avviare un aggiornamento blu/verde in produzione.
- È preferibile effettuare il passaggio dal server blu a quello verde durante la finestra di manutenzione.
- Per garantire che tutto funzioni correttamente dopo l'aggiornamento e la sincronizzazione, è consigliabile conservare il cluster originale per un certo periodo di tempo prima di eliminarlo. Potrebbe rivelarsi utile in caso di problemi imprevisti.
- Evita operazioni di scrittura pesanti come i caricamenti in blocco durante l'esecuzione della soluzione blu/verde Neptune, poiché possono causare ritardi di replica che introducono tempi di inattività significativi. Idealmente, il tempo che intercorre tra la disattivazione delle scritture sul cluster blu e l'attivazione per il cluster verde è di pochi istanti.

Risoluzione dei problemi relativi alla soluzione blu/verde Neptune

Errori generati dalla soluzione blu/verde Neptune

- **`Cluster with id = (blue_green_deployment_id) already exists`**: è presente un cluster esistente con identificatore *`(blue_green_deployment_id)`*.

  Specifica un nuovo ID di implementazione o imposta la modalità di implementazione su `resume` se il cluster è stato creato in un'esecuzione blu/verde Neptune precedente.
- **`Streams should be enabled on the source Cluster for Blue Green Deployment`**: abilita i flussi Neptune nel cluster blu (origine).
- **`No Bulkload should be in progress on source cluster: (cluster_id)`**: la soluzione blu/verde Neptune termina se rileva un caricamento in blocco in corso.

  Questo per garantire che il processo di sincronizzazione possa recuperare il ritardo con le scritture effettuate. Evita o annulla qualsiasi operazione di caricamento in blocco in corso prima di avviare la soluzione blu/verde Neptune.
- **`Blue Green deployment requires instances to be in sync with db cluster parameter group`**: qualsiasi modifica al gruppo di parametri del cluster deve essere sincronizzata in tutto il cluster database. Per informazioni, consultare Gruppi di parametri di Amazon Neptune.
- **`Invalid target engine version for Blue Green Deployment`**: il rilascio del motore di destinazione deve essere elencato come attivo in Rilasci del motore per Amazon Neptune e deve essere successivo al rilascio corrente del motore del cluster di origine (blu).

Creazione di un utente IAM con autorizzazioni per Neptune

Per accedere alla console Neptune al fine di creare e gestire un cluster database Neptune, devi creare un utente IAM con tutte le autorizzazioni necessarie.

La prima fase è creare una policy di ruolo collegato ai servizi per Neptune:

Crea una policy di ruolo collegato ai servizi per Amazon Neptune

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione a sinistra, seleziona Policies (Policy).
3. Nella pagina Policy, seleziona Crea policy.
4. Nella pagina Crea policy, seleziona la scheda JSON e copia la seguente policy di ruolo collegato ai servizi:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
      "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"rds.amazonaws.com"
        }
      }
    }
  ]
}
```

5. Seleziona Successivo: Tag e nella pagina Aggiungi tag seleziona Successivo: Rivedi.
6. Nella pagina Verifica policy, assegna alla nuova policy il nome "NeptuneServiceLinked".

Per ulteriori informazioni sui ruoli collegati al servizio, consulta Utilizzo di ruoli collegati ai servizi per Neptune.

Crea un nuovo utente IAM con tutte le autorizzazioni necessarie

Poi crea il nuovo utente IAM collegando le policy gestite appropriate che concederanno le autorizzazioni di cui avrai bisogno, insieme alla policy di ruolo collegato ai servizi che hai creato (denominata `NeptuneServiceLinked`):

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione a sinistra, scegli Utenti e nella pagina Utenti scegli Aggiungi utenti.
3. Nella pagina Aggiungi utente, inserisci un nome per il nuovo utente IAM, scegli Chiave di accesso - Accesso programmatico per il tipo di credenziale AWS e scegli Successivo: Autorizzazioni.
4. Nella pagina Imposta autorizzazioni, nella casella Policy di filtro, digita "Neptune". Ora seleziona quanto segue dalle policy elencate:

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked (supponendo che sia questo il nome assegnato alla policy di ruolo collegata al servizio creata in precedenza).
5. Poi digita "VPC" nella casella Filtra policy al posto di "Neptune". Seleziona AmazonVPCFullAccess dalle policy elencate.
6. Seleziona Successivo: Tag e nella pagina Aggiungi tag seleziona Successivo: Rivedi.
7. Nella pagina Verifica, controlla che tutte le seguenti policy siano ora allegate al nuovo utente:

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked
   - AmazonVPCFullAccess

   Seleziona Crea utente.
8. Infine, scarica e salva l'ID chiave di accesso e la chiave di accesso segreta del nuovo utente.

Per l'interoperabilità con altri servizi come Amazon Simple Storage Service (Amazon S3), è necessario aggiungere autorizzazioni e relazioni di trust.

Gruppi di parametri di Amazon Neptune

Puoi gestire la configurazione del database in Amazon Neptune utilizzando i parametri in un gruppo di parametri. I gruppi di parametri fungono da container per i valori di configurazione di un motore applicati a una o più istanze database.

Sono disponibili due tipi di gruppi di parametri: gruppi di parametri del cluster database e gruppi di parametri database.

- I gruppi di parametri database si applicano a livello di istanza e in genere sono associati alle impostazioni per il motore a grafo di Neptune, ad esempio il parametro `neptune_query_timeout`.
- I gruppi di parametri del cluster database vengono applicati a tutte le istanze del cluster e generalmente hanno una gamma di impostazioni più ampia. Ogni cluster Neptune è associato a un gruppo di parametri del cluster database. Ogni istanza database all'interno del cluster eredita i valori di configurazione del motore contenuti nel gruppo di parametri del cluster database.

Eventuali valori di configurazione modificati nel gruppo di parametri del cluster database sostituiscono i valori predefiniti nel gruppo di parametri database. Se modifichi i valori corrispondenti nel gruppo di parametri database, tali valori sostituiranno le impostazioni nel gruppo di parametri del cluster database.

Un gruppo di parametri database predefinito viene utilizzato se crei un'istanza database senza specificare un gruppo di parametri database personalizzato. Non puoi modificare le impostazioni dei parametri del gruppo di parametri database predefinito. Per modificare le impostazioni dei parametri predefiniti devi creare un nuovo gruppo di parametri database. Non tutti i parametri del motore di database possono essere modificati nel gruppo di parametri database che hai creato.

I gruppi di parametri vengono creati in famiglie compatibili con diverse versioni del motore Neptune. La famiglia del gruppo di parametri predefinito è `neptune1`, che è compatibile con tutte le versioni del motore precedenti alla `1.2.0.0`. A partire dalla Rilascio: 1.2.0.0 (21/07/2022), è necessario utilizzare invece la famiglia del gruppo di parametri `neptune1.2`. Ciò significa che quando si esegue l'aggiornamento alla versione `1.2.0.0` o superiore, è necessario innanzitutto ricreare tutti i gruppi di parametri personalizzati nella famiglia `neptune1.2` in modo da poterli collegare durante l'aggiornamento.

Alcuni parametri di Neptune sono statici e altri dinamici. Le differenze sono le seguenti:

Parametri statici

- Un parametro statico ha effetto solo dopo il riavvio di un'istanza database. In altre parole, quando modifichi un parametro statico e salvi il gruppo di parametri database dell'istanza, devi riavviare manualmente l'istanza database affinché la modifica del parametro venga applicata. Attualmente, tutti i parametri a livello di istanza di Neptune (in un gruppo di parametri database anziché in un gruppo di parametri del cluster database) sono statici.
- Quando modifichi un parametro statico a livello di cluster e salvi il gruppo di parametri del cluster database, la modifica del parametro viene applicata dopo aver riavviato manualmente ogni istanza database sul cluster.

Parametri dinamici

- Un parametro dinamico viene applicato quasi immediatamente dopo essere stato aggiornato nel rispettivo gruppo di parametri. In altre parole, non è necessario riavviare un'istanza database dopo aver aggiornato un parametro dinamico per applicarne la modifica.
- È previsto un lieve ritardo per l'applicazione in tutte le istanze database della modifica di un parametro dinamico del cluster.
- Un valore di parametro dinamico aggiornato non viene applicato alle richieste attualmente in esecuzione, ma solo a quelle inviate dopo la modifica.
- Quando si modifica un parametro dinamico a livello di cluster, per impostazione predefinita la modifica del parametro viene applicata immediatamente al cluster database, senza che sia necessario il riavvio. Per rinviare la modifica del parametro a dopo il riavvio delle istanze database nel cluster, utilizza AWS CLI per impostare `ApplyMethod` su `pending-reboot` per la modifica del parametro.

Attualmente tutti i parametri sono statici ad eccezione dei seguenti nuovi parametri del cluster:

- `neptune_enable_slow_query_log` (a livello di cluster)
- `neptune_slow_query_log_threshold` (a livello di cluster)

Di seguito sono elencati alcuni punti importanti sull'utilizzo dei parametri in un gruppo di parametri database:

- Un'impostazione errata dei parametri in un gruppo di parametri database può avere conseguenze negative impreviste, tra cui il peggioramento delle prestazioni e l'instabilità del sistema. Fai sempre attenzione quando modifichi i parametri database ed effettui il backup dei dati prima di modificare

un gruppo di parametri database. Prova le modifiche delle impostazioni del gruppo di parametri su un'istanza database di test prima di applicare le modifiche apportate a un'istanza database di produzione.

- Quando modifichi il gruppo di parametri database associato a un'istanza database, devi riavviare manualmente l'istanza prima che il nuovo gruppo di parametri database venga usato dall'istanza database.

> **Note**
>
> Prima della Rilascio: 1.2.0.0 (21/07/2022), tutte le istanze di replica di lettura in un cluster database venivano riavviate automaticamente ogni volta che l'istanza primaria (writer) veniva riavviata.
> A partire dalla Rilascio: 1.2.0.0 (21/07/2022), il riavvio dell'istanza primaria non causa il riavvio di nessuna delle istanze di replica. Ciò significa che, se modifichi un parametro a livello di cluster, devi riavviare ogni istanza separatamente per riprendere la modifica del parametro.

Modifica di un gruppo di parametri del cluster di database o di un gruppo di parametri di database

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Scegliere Parameter groups (Gruppi di parametro) nel riquadro di navigazione.
3. Scegliere il collegamento Name (Nome) per il gruppo di parametri database che desideri modificare.

   (Facoltativo) Scegliere Create parameter group (Crea gruppo di parametri) per creare un nuovo gruppo di parametri del cluster e crea il nuovo gruppo. Scegli il Name (Nome) del nuovo gruppo di parametri.

   > **Important**
   >
   > Questo passaggio è richiesto se disponi solo del gruppo di parametri del cluster di database predefinito perché il gruppo di parametri del cluster di database predefinito non può essere modificato.

4. Cercate il parametro e fate clic sul campo Valore accanto alla colonna Nome.
5. Inserisci il valore consentito e scegli il segno di spunta accanto al campo del valore.
6. Seleziona Salvataggio delle modifiche.
7. Riavvia ogni istanza database nel cluster Neptune se stai modificando un parametro del cluster database oppure riavvia una o più istanze specifiche se stai modificando un parametro di istanza database.

Creazione di un gruppo di parametri database o di un gruppo di parametri del cluster database

Puoi utilizzare la console Neptune per creare facilmente un nuovo gruppo di parametri:

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Scegliere Parameter groups (Gruppi di parametro) nel riquadro sinistro di navigazione.
3. Scegliere Create DB parameter group (Crea un gruppo di parametri database).

   Compare la pagina Create DB parameter group (Crea gruppo di parametri di database).
4. Nell'elenco Famiglia del gruppo di parametri, scegli neptune1 oppure, se hai come target il motore versione 1.2.0.0 o successiva, scegli neptune1.2.
5. Nell'elenco Type (Tipo), scegliere DB Parameter Group (Gruppo di parametri database) o DB Cluster Parameter Group (Gruppo di parametri del cluster di database).
6. Nella casella Group name (Nome gruppo), digita il nome del nuovo gruppo di parametri database.
7. Nella casella Description (Descrizione), digita una descrizione per il nuovo gruppo di parametri database.
8. Scegli Crea.

Puoi creare un nuovo gruppo di parametri anche utilizzando AWS CLI:

```
aws neptune create-db-parameter-group \
  --db-parameter-group-name (a name for the new DB parameter group) \
  --db-parameter-group-family (either neptune1 or neptune1.2, depending on the engine
 version) \
  --description (a description for the new DB parameter group)
```

Parametri di Amazon Neptune

La configurazione del database in Amazon Neptune viene gestita utilizzando i parametri nei gruppi di parametri. Sono disponibili i seguenti parametri per configurare il database Neptune:

Parametri a livello di cluster

- neptune_enable_audit_log
- neptune_enable_slow_query_log
- neptune_slow_query_log_threshold
- neptune_lab_mode
- neptune_query_timeout
- neptune_streams
- neptune_streams_expiry_days
- neptune_lookup_cache
- neptune_autoscaling_config
- neptune_ml_iam_role
- neptune_ml_endpoint

Parametri a livello di istanza

- neptune_dfe_query_engine
- neptune_query_timeout
- neptune_result_cache

Parametri obsoleti

- neptune_enforce_ssl

neptune_enable_audit_log (parametro a livello di cluster)

Questo parametro attiva o disattiva la registrazione dei log di audit per Neptune.

I valori consentiti sono `0` (disabilitati) e `1` (abilitati). Il valore predefinito è `0`.

Questo parametro è statico, il che significa che eventuali modifiche non hanno effetto su nessuna istanza finché non viene riavviata.

Puoi pubblicare i log di audit su Amazon CloudWatch, come descritto in Utilizzo della CLI per pubblicare i log di controllo di Neptune su Logs CloudWatch .

`neptune_enable_slow_query_log` (parametro a livello di cluster)

Utilizza questo parametro per abilitare o disabilitare la funzionalità di registrazione di log delle query lente di Neptune.

Si tratta di un parametro dinamico, il che significa che la modifica del suo valore non richiede né causa il riavvio del cluster database.

I valori consentiti sono:

- **`info`**: abilita la registrazione di log delle query lente e registra gli attributi selezionati che potrebbero contribuire al rallentamento delle prestazioni.
- **`debug`**: abilita la registrazione di log delle query lente e registra tutti gli attributi disponibili dell'esecuzione della query.
- **`disable`**: disabilita la registrazione di log delle query lente.

Il valore predefinito è `disable`.

Puoi pubblicare i log delle query lente su Amazon CloudWatch, come descritto in Utilizzo della CLI per pubblicare i log di Neptune con query lente su Logs CloudWatch .

`neptune_slow_query_log_threshold` (parametro a livello di cluster)

Questo parametro specifica la soglia del tempo di esecuzione, in millisecondi, dopo la quale una query viene considerata lenta. Se la registrazione di log delle query lente è abilitata, le query che durano più a lungo di questa soglia verranno registrate insieme ad alcuni dei loro attributi.

Il valore predefinito è 5000 millisecondi (5 secondi).

Si tratta di un parametro dinamico, il che significa che la modifica del suo valore non richiede né causa il riavvio del cluster database.

`neptune_lab_mode` (parametro a livello di cluster)

Quando impostato, questo parametro abilita funzionalità sperimentali specifiche di Neptune. Vedere Modalità di laboratorio Neptune per le funzionalità sperimentali attualmente disponibili.

Questo parametro è statico, il che significa che eventuali modifiche non hanno effetto su nessuna istanza finché non viene riavviata.

Per abilitare o disabilitare una funzionalità sperimentale, includere *(nome funzione)*=enabled o *(nome funzione)* =disabled in questo parametro. È possibile abilitare o disabilitare più funzionalità separandole con virgole, in questo modo:

(nome funzione #1)=enabled,*(nome funzione #2)*=enabled

Le funzionalità della modalità Lab sono in genere disabilitate per impostazione predefinita. Un'eccezione è la funzionalità DFEQueryEngine, che è stata abilitata per impostazione predefinita in modo da essere usata con i suggerimenti di query (DFEQueryEngine=viaQueryHint) a partire dalla versione 1.0.5.0 del motore Neptune. A partire dal rilascio 1.1.1.0 del motore Neptune, il motore DFE non è più in modalità Lab e ora viene controllato con il parametro di istanza neptune_dfe_query_engine nel gruppo di parametri database di un'istanza.

`neptune_query_timeout` (parametro a livello di cluster)

Indica una durata di timeout specifica per le query dei grafi, espressa in millisecondi.

I valori consentiti vanno da 10 a 2,147,483,647 (2^{31} - 1). Il valore predefinito è 120,000 (2 minuti).

Questo parametro è statico, il che significa che eventuali modifiche non hanno effetto su nessuna istanza finché non viene riavviata.

> **Note**
>
> Potrebbero essere applicati costi imprevisti se viene impostato un valore di timeout delle query troppo elevato, in particolare su un'istanza serverless. Senza il timeout non è impostato su un valore ragionevole, potresti inavvertitamente generare una query che continua a essere eseguita molto più a lungo del previsto, con conseguenti costi imprevisti. Questo è particolarmente vero per un'istanza serverless, che potrebbe aumentare fino a diventare un tipo di istanza di grandi dimensioni e costosa durante l'esecuzione della query.

Per evitare spese impreviste di questo tipo, specifica un valore di timeout delle query adatto alla maggior parte delle query e che causa il timeout solo delle query con tempi di esecuzione inaspettatamente lunghi.

`neptune_streams` (parametro a livello di cluster)

Abilita o disabilita Neptune Streams.

Questo parametro è statico, il che significa che eventuali modifiche non hanno effetto su nessuna istanza finché non viene riavviata.

I valori consentiti sono `0` (disabilitati, che è l'impostazione predefinita) e `1` (abilitati).

`neptune_streams_expiry_days` (parametro a livello di cluster)

Specifica quanti giorni devono trascorrere prima che il server elimini i record di flusso.

I valori validi vanno da `1` a `90`, incluso. Il valore predefinito è `7`.

Questo parametro è stato introdotto nella versione 1.2.0.0 del motore.

Questo parametro è statico, il che significa che eventuali modifiche non hanno effetto su nessuna istanza finché non viene riavviata.

`neptune_lookup_cache` (parametro a livello di cluster)

Disabilita o riabilita la cache di ricerca di Neptune. sulle istanze `R5d`.

Questo parametro è statico, il che significa che eventuali modifiche non hanno effetto su nessuna istanza finché non viene riavviata.

I valori consentiti sono `enabled` e `disabled`. Il valore predefinito è `disabled`, ma ogni volta che viene creata un'istanza `R5d` nel cluster database, il parametro `neptune_lookup_cache` viene impostato automaticamente su `enabled` e sull'istanza viene creata una cache di ricerca.

`neptune_autoscaling_config` (parametro a livello di cluster)

Imposta i parametri di configurazione per le istanze di replica di lettura create e gestite dal dimensionamento automatico di Neptune.

Questo parametro è statico, il che significa che eventuali modifiche non hanno effetto su nessuna istanza finché non viene riavviata.

Utilizza una stringa JSON impostata come valore del parametro `neptune_autoscaling_config` per specificare:

- Il tipo di istanza utilizzato dal dimensionamento automatico di Neptune per tutte le nuove istanze di replica di lettura che crea.
- Le finestre di manutenzione assegnate alle repliche di lettura.
- I tag da associare a tutte le nuove repliche di lettura.

La stringa JSON ha una struttura simile alla seguente:

```
"{
  \"tags\": [
    { \"key\" : \"reader tag-0 key\", \"value\" : \"reader tag-0 value\" },
    { \"key\" : \"reader tag-1 key\", \"value\" : \"reader tag-1 value\" },
  ],
  \"maintenanceWindow\" : \"wed:12:03-wed:12:33\",
  \"dbInstanceClass\" : \"db.r5.xlarge\"
}"
```

Nota che tutte le virgolette all'interno della stringa devono essere precedute dal carattere di escape della barra rovesciata (\).

Tutte le tre impostazioni di configurazione non specificate nel parametro `neptune_autoscaling_config` vengono copiate dalla configurazione dell'istanza writer primaria del cluster database.

`neptune_ml_iam_role` (parametro a livello di cluster)

Specifica l'ARN del ruolo IAM utilizzato in Neptune ML. Il valore può essere qualsiasi ARN valido del ruolo IAM.

Questo parametro è statico, il che significa che eventuali modifiche non hanno effetto su nessuna istanza finché non viene riavviata.

Puoi specificare l'ARN predefinito del ruolo IAM per il machine learning sui grafi.

`neptune_ml_endpoint` (parametro a livello di cluster)

Specifica l'endpoint utilizzato per Neptune ML. Il valore può essere qualsiasi nome di endpoint SageMaker valido.

Questo parametro è statico, il che significa che eventuali modifiche non hanno effetto su nessuna istanza finché non viene riavviata.

È possibile specificare l'endpoint SageMaker predefinito per il machine learning sui grafi.

`neptune_dfe_query_engine` (parametro a livello di istanza)

A partire dalla versione 1.1.1.0 del motore Neptune, questo parametro di istanza database viene utilizzato per controllare l'utilizzo del motore di query DFE. I valori consentiti sono i seguenti:

Questo parametro è statico, il che significa che eventuali modifiche non hanno effetto su nessuna istanza finché non viene riavviata.

- **`enabled`**: fa sì che il motore DFE venga utilizzato ogni volta che è possibile, tranne nei casi in cui il suggerimento di query `useDFE` esiste ed è impostato su `false`.
- **`viaQueryHint`** (impostazione predefinita): fa sì che il motore DFE venga utilizzato solo per le query che includono esplicitamente il suggerimento di query `useDFE` impostato su `true`.

Se questo parametro non è stato impostato in modo esplicito, all'avvio dell'istanza viene utilizzato il valore predefinito `viaQueryHint`.

> **Note**
>
> Tutte le query openCypher vengono eseguite dal motore DFE a prescindere dall'impostazione di questo parametro.

Prima della versione 1.1.1.0, questo era un parametro in modalità Lab anziché un parametro di istanza database.

`neptune_query_timeout` (parametro a livello di istanza)

Questo parametro di istanza database specifica la durata del timeout (in millisecondi) delle query sui grafi per una singola istanza.

Questo parametro è statico, il che significa che eventuali modifiche non hanno effetto su nessuna istanza finché non viene riavviata.

I valori consentiti vanno da `10` a `2,147,483,647` (2^{31} - 1). Il valore predefinito è `120,000` (2 minuti).

> **Note**
>
> Potrebbero essere applicati costi imprevisti se viene impostato un valore di timeout delle query troppo elevato, in particolare su un'istanza serverless. Senza il timeout non è impostato su un valore ragionevole, potresti inavvertitamente generare una query che continua a essere eseguita molto più a lungo del previsto, con conseguenti costi imprevisti. Questo è particolarmente vero per un'istanza serverless, che potrebbe aumentare fino a diventare un tipo di istanza di grandi dimensioni e costosa durante l'esecuzione della query.
> Per evitare spese impreviste di questo tipo, specifica un valore di timeout delle query adatto alla maggior parte delle query e che causa il timeout solo delle query con tempi di esecuzione inaspettatamente lunghi.

`neptune_result_cache` (parametro a livello di istanza)

`neptune_result_cache`: questo parametro dell'istanza database abilita o disabilita i Memorizzazione nella cache dei risultati delle query.

Questo parametro è statico, il che significa che eventuali modifiche non hanno effetto su nessuna istanza finché non viene riavviata.

I valori consentiti sono `0` (disabilitati, impostazione predefinita) e `1` (abilitati).

`neptune_enforce_ssl` (parametro a livello di cluster OBSOLETO)

(Obsoleto) In passato esistevano regioni che consentivano le connessioni HTTP a Neptune e questo parametro, se impostato su 1, veniva utilizzato per forzare tutte le connessioni a utilizzare HTTPS. Questo parametro non è più utilizzato dato che Neptune ora accetta solo connessioni HTTPS in tutte le regioni.

Avvio di un cluster database Neptune mediante la console AWS Management Console

Il modo più semplice per avviare un nuovo cluster database Neptune consiste nell'utilizzare un modello AWS CloudFormation che crei tutte le risorse necessarie, come spiegato in Creazione di un cluster DB.

Se preferisci, puoi anche usare la console Neptune per avviare manualmente un nuovo cluster database, come spiegato qui.

Per accedere alla console Neptune al fine di creare e gestire un cluster database Neptune, devi creare un utente IAM con tutte le autorizzazioni necessarie, come spiegato in Creazione di un utente IAM con autorizzazioni per Neptune.

Poi accedi alla AWS Management Console come utente IAM e procedi come descritto di seguito per creare un nuovo cluster database:

Per avviare un cluster database Neptune mediante la console

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Vai alla pagina Database e scegli Crea database. Verrà aperta la pagina Crea database.
3. In Opzioni del motore, il tipo di motore è `neptune`. Puoi scegliere una versione specifica del motore o accettare quella predefinita.
4. In Impostazioni, inserisci un nome per il nuovo cluster database o accetta il nome predefinito che viene fornito. Questo nome viene utilizzato nell'indirizzo dell'endpoint dell'istanza e deve soddisfare i seguenti vincoli:
   - Deve contenere da 1 a 63 caratteri alfanumerici o trattini.
   - Il primo carattere deve essere una lettera.
   - Non può terminare con un trattino o contenere due trattini consecutivi.
   - Deve essere univoco in tutte le istanze database nell'account AWS e in una determinata regione AWS.
5. In Modelli, scegli Produzione o Sviluppo e test.
6. In Dimensione dell'istanza database, scegli una dimensione dell'istanza che determina la capacità di elaborazione e memoria dell'istanza di scrittura principale del nuovo cluster database.

Se hai selezionato il modello Produzione, puoi scegliere solo tra le classi ottimizzate per la memoria disponibili ed elencate. Se, invece, hai selezionato Sviluppo e test, puoi anche scegliere tra le classi espandibili più economiche (vedi Istanze espandibili T3 per una panoramica delle classi espandibili).

> **Note**
>
> A partire dal rilascio 1.1.0.0 del motore Neptune, i tipi di istanza R4 non sono più supportati.

7. In Disponibilità e durata, puoi scegliere se abilitare o meno l'implementazione in più zone di disponibilità (Multi-AZ). Il modello Produzione abilita l'implementazione multi-AZ per impostazione predefinita, mentre il modello Sviluppo e test no. Se l'implementazione multi-AZ è abilitata, Neptune individua le istanze di replica di lettura create dall'utente in diverse zone per migliorare la disponibilità.
8. In Connettività, seleziona tra le scelte disponibili il cloud privato virtuale (VPC) che ospiterà il nuovo cluster database. Qui puoi scegliere Crea nuovo VPC se vuoi che Neptune crei il VPC automaticamente. Devi creare un'istanza Amazon EC2 nello stesso VPC per accedere all'istanza Neptune (per ulteriori informazioni, consulta Ogni cluster database Amazon Neptune risiede in un Amazon VPC). Non puoi modificare il VPC dopo la creazione del cluster database.

   Se necessario, puoi configurare ulteriormente la connettività per il tuo cluster in Configurazione di connettività aggiuntiva:

   a. In Gruppo di sottoreti, puoi scegliere il gruppo di sottoreti di database Neptune da utilizzare per il nuovo cluster database. Se il tuo VPC non contiene gruppi di sottoreti, Neptune ne crea uno automaticamente (vedi Ogni cluster database Amazon Neptune risiede in un Amazon VPC).
   b. In Gruppi di sicurezza VPC, scegli uno o più gruppi di sicurezza VPC esistenti per proteggere l'accesso di rete al nuovo cluster database. In alternativa, scegli Crea nuovo se desideri che Neptune ne crei uno per te, poi fornisci un nome per il nuovo gruppo di sicurezza VPC (vedi Creare un gruppo di sicurezza tramite la console VPC).
   c. In Porta del database, inserisci la porta TCP/IP che il database utilizzerà per le connessioni alle applicazioni. Neptune utilizza il numero di porta 8182 per impostazione predefinita.
9. In Configurazione notebook, scegli Crea notebook se desideri che Neptune crei automaticamente notebook Jupyter nell'ambiente Neptune (vedi Utilizzo di notebook per grafi

Neptune per iniziare rapidamente e Utilizzo Neptune Workbench per ospitare i notebook Neptune). Puoi quindi scegliere come configurare i nuovi notebook:

a. In Tipo di istanza notebook, scegli tra le classi di istanza disponibili per il tuo notebook.

b. In Nome notebook, inserisci un nome per il notebook.

c. Se lo desideri, puoi anche inserire una descrizione del notebook in Descrizione - facoltativo.

d. In Nome ruolo IAM, scegli se lasciare che sia Neptune a creare un ruolo IAM per il notebook, poi inserisci un nome per il nuovo ruolo. In alternativa, scegli di selezionare un ruolo IAM esistente tra quelli disponibili.

e. Infine, scegli se il notebook deve connettersi a Internet direttamente oppure tramite Amazon SageMaker o un VPC con un gateway NAT. Per ulteriori informazioni, consulta Connessione di un'istanza notebook alle risorse in un VPC.

10. In Tag, puoi associare fino a un massimo di 50 tag al nuovo cluster database.

11. In Configurazione aggiuntiva ci sono altre impostazioni che puoi configurare per il nuovo cluster database (in molti casi, puoi saltarle e accettare i valori predefiniti, almeno per ora):

Opzione	Cosa puoi fare
DB instance identifier (Identificatore istanze DB)	Puoi fornire un nome per l'istanza writer del cluster. Se decidi di non farlo, viene utilizzato un identificatore predefinito basato sul nome del cluster. Se vuoi farlo, specifica un nome univoco per tutte le istanze database appartenenti al tuo account AWS nella regione corrente. L'identificatore di istanza database non fa distinzione tra maiuscole e minuscole, ma viene archiviato solo in minuscolo.
DB cluster parameter group (Gruppo di parametri del cluster database)	Seleziona un gruppo di parametri del cluster database per definire la configurazione predefinita per tutte le istanze database del cluster. A meno tu non scelga diversamente, Neptune utilizza un gruppo di parametri del cluster database predefinito. Per ulteriori

Opzione	Cosa puoi fare
	informazioni sui gruppi di parametri, consultar e Gruppi di parametri di Amazon Neptune.
DB parameter group (Gruppo di parametri database)	Seleziona un gruppo di parametri del database per definire la configurazione dell'istanza database primaria nel cluster. A meno tu non scelga diversamente, Neptune utilizza un gruppo di parametri predefini to. Per ulteriori informazioni sui gruppi di parametri, consultare Gruppi di parametri.
IAM DB authentication (Autenticazione DB IAM)	Se selezioni Abilita autenticazione DB IAM, tutti gli accessi al database verranno autenticati tramite AWS Identity and Access Management (IAM). **Important** Ciò richiede la firma di tutte le richieste con AWS Signature versione 4. Per ulteriori informazioni, consulta Panoramica di AWS Identity and Access Management (IAM) in Amazon Neptune.
Failover priority (Priorità failover)	Scegli `No preference` un livello di priorità per il failover. Se scegli un livello e si verifica un conflitto al suo interno, viene selezionata la replica con le stesse dimensioni dell'ista nza principale.

Opzione	Cosa puoi fare
Backup retention period (Periodo di retention dei backup)	Seleziona l'intervallo di tempo, da 1 a 35 giorni, durante il quale Neptune deve conservare i backup automatici di questa istanza database. Puoi eseguire un ripristino point-in-time (PITR) solo entro il periodo di conservazione dei backup.
Copy tags to snapshots (Copia tag in snapshot)	(Attivata per impostazione predefinita) Questa opzione fa sì che tutti i tag associati al cluster database vengano copiati in qualsiasi snapshot del cluster.
Abilita crittografia	(Attivata per impostazione predefinita) Questa opzione fa sì che i dati nel cluster database vengano crittografati quando sono inattivi. Se decidi di farlo, scegli la chiave master utilizzata per proteggere la chiave impiegata per crittografare questo volume di database. Puoi selezionare la chiave predefinita `aws/rds` oppure scegliere tra le chiavi master presenti nel tuo account o inserire l'ARN di una chiave di un altro account. Puoi creare una nuova chiave di crittografia master nella scheda Chiavi di crittografia della console IAM. Per ulteriori informazioni, consulta Crittografia delle risorse Neptune inattive.
Log di audit	Seleziona questa opzione se desideri che i log di audit del cluster database vengano pubblicati su CloudWatch Logs.

Opzione	Cosa puoi fare
Abilita aggiornamento automatico della versione secondaria	(Attivata per impostazione predefinita) Questa opzione fa sì che il cluster database venga aggiornato automaticamente alle nuove versioni secondarie del motore dopo il loro rilascio. Gli aggiornamenti automatici vengono eseguiti durante la finestra di manutenzione del database. Per informazioni, consultare Uso di `AutoMinorVersionUpgrade`.
Maintenance window (Finestra di manutenzione)	Puoi selezionare un periodo specifico durante il quale apportare le modifiche in sospeso al cluster database, ad esempio una modifica a una classe di istanza database oppure una patch automatica del motore. Tutte le operazioni di manutenzione di questo tipo vengono avviate e completate entro il periodo selezionato. Se non selezioni un periodo, Neptune assegna un periodo di manutenzione in modo arbitrario.
Enable deletion protection (Abilita protezione da eliminazione)	(EnAttivata per impostazione predefinita) La protezione da eliminazione impedisce l'eliminazione del cluster database. Devi disabilitarla in modo esplicito per poter eliminare il cluster database.

12. Scegli Crea database per avviare il nuovo cluster database Neptune e la rispettiva istanza principale.

    Nella console Amazon Neptune, il nuovo cluster database viene visualizzato nell'elenco dei database. Lo stato del cluster di database è Creating (Creazione in corso) fino a quando non viene creato ed è pronto per l'uso. Quando lo stato passa su Available (Disponibile), puoi connetterti all'istanza principale per il cluster database. A seconda della classe dell'istanza database e dello store allocato, prima che le nuove istanze database siano disponibili potrebbero trascorrere diversi minuti.

Per visualizzare il cluster appena creato, scegli la vista Database nella console Neptune.

> **Note**
>
> Se elimini tutte le istanze database di Neptune in un cluster database utilizzando AWS Management Console, la console elimina automaticamente il cluster database stesso. Se utilizzi AWS CLI oppure l'SDK, devi eliminare il cluster database manualmente dopo aver eliminata l'ultima istanza al suo interno.

Prendi nota del valore di Endpoint cluster. Servirà per la connessione al cluster database Neptune.

Arresto e avvio di un cluster database Amazon Neptune

L'avvio e l'arresto dei cluster Amazon Neptune è utile per gestire i costi degli ambienti di test e sviluppo. Puoi arrestare temporaneamente tutte le istanze database nel cluster invece di impostare e rimuovere tutte le istanze database ogni volta che utilizzi il cluster.

Argomenti

Panoramica dell'avvio e dell'arresto di un cluster database Neptune

Nei periodi in cui non hai bisogno di un cluster Neptune, puoi arrestare contemporaneamente tutte le istanze del cluster. Puoi avviare nuovamente il cluster ogni volta che devi utilizzarlo. L'avvio e l'arresto semplificano i processi di impostazione e rimozione dei cluster utilizzati per lo sviluppo, i test o attività simili che non richiedono una disponibilità continua. Puoi eseguire questa operazione nella AWS Management Console con una singola azione, indipendentemente dal numero di istanze presenti nel cluster.

Per tutta la durata della sospensione di un cluster di database, vengono addebitati solo i costi per lo storage del cluster, gli snapshot manuali e lo storage di backup automatici all'interno della finestra di retention specificata. Non è previsto alcun addebito per le ore dell'istanza database.

Dopo sette giorni, Neptune riavvia automaticamente il cluster database in modo da non perdere gli aggiornamenti di manutenzione richiesti.

Per ridurre gli addebiti per un cluster Neptune con poco carico, puoi arrestarlo anziché eliminarne tutte le repliche. Per i cluster che hanno più di una o due istanze, eliminare frequentemente le istanze database e ricrearle è pratico solo se si utilizza AWS CLI o l'API Neptune. Inoltre, le eliminazioni possono essere difficili da eseguire nell'ordine corretto. Ad esempio, è necessario eliminare tutte le repliche di lettura prima di eliminare l'istanza primaria per evitare di attivare il meccanismo di failover.

Non utilizzare l'avvio e l'arresto se è necessario mantenere il cluster DB in esecuzione, ma con capacità ridotta. Se il cluster è troppo costoso o non molto impiegato, puoi eliminare una o più istanze DB o modificare le istanze DB per utilizzare una classe di istanza più piccola, ma non puoi arrestare una singola istanza DB.

Arresto di un cluster database Neptune

Se non è utilizzato, è possibile arrestare un cluster database Neptune in esecuzione e quindi avviarlo di nuovo quando necessario. Per tutta la durata dell'arresto del cluster, vengono addebitati i costi per lo storage del cluster, gli snapshot manuali e lo storage di backup automatici all'interno della finestra di retention specificata, ma non per le ore di utilizzo dell'istanza database.

L'operazione di arresto interrompe tutte le istanze di replica di lettura del cluster prima di arrestare l'istanza primaria, per evitare di attivare il meccanismo di failover.

Arresto di un cluster database utilizzando la AWS Management Console

Per utilizzare la AWS Management Console per arrestare un cluster Neptune

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel pannello di navigazione, scegliere Databases (Database), quindi scegliere un cluster. Si può eseguire l'operazione di arresto da questa pagina o andare alla pagina dei dettagli del cluster di database da arrestare.
3. In Actions (Operazioni) scegliere Stop (Arresta).

Arresto di un cluster database utilizzando la AWS CLI

Per arrestare un'istanza DB utilizzando l'AWS CLI, chiama il comando stop-db-cluster utilizzando il parametro `--db-cluster-identifier` per identificare il cluster DB che vuoi arrestare.

Example

```
aws neptune stop-db-cluster --db-cluster-identifier mydbcluster
```

Arresto di un cluster database utilizzando l'API di gestione Neptune

Per arrestare un'istanza database utilizzando l'API di gestione Neptune, chiama l'API StopDBCluster e utilizza il parametro `DBClusterIdentifier` per identificare il cluster database che vuoi arrestare.

Cosa può accadere mentre un cluster database viene arrestato

- È possibile ripristinarlo da una snapshot (consulta Ripristino da una snapshot del cluster di database).

- Non è possibile modificare la configurazione del cluster DB o di una delle sue istanze database.
- Non è possibile aggiungere o rimuovere istanze database dal cluster.
- Non è possibile eliminare il cluster se ci sono ancora istanze database associate.
- In generale, è necessario riavviare un cluster DB arrestato per eseguire la maggior parte delle operazioni amministrative.
- Neptune applica la manutenzione pianificata al cluster arrestato dopo il suo riavvio. Tieni presente che dopo sette giorni, Neptune riavvia automaticamente un cluster arrestato in modo che non accumuli un ritardo eccessivo nell'esecuzione delle operazioni di manutenzione.
- Neptune non esegue alcun backup automatico di un cluster database arrestato, perché i dati sottostanti non possono cambiare mentre il cluster è arrestato.
- Neptune non estende il periodo di conservazione del backup per il cluster database mentre è arrestato.

Avvio di un cluster database Neptune arrestato

È possibile avviare solo un cluster database Neptune che si trova in stato arrestato. Quando avvii il cluster, tutte le sue istanze database tornano disponibili. Il cluster conserva le sue impostazioni di configurazione, come gli endpoint, i gruppi di parametri e i gruppi di sicurezza VPC.

Avvio di un cluster database arrestato utilizzando la AWS Management Console

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel pannello di navigazione, scegliere Databases (Database), quindi scegliere un cluster. È possibile eseguire l'operazione di avvio da questa pagina o andare alla pagina dei dettagli del cluster DB e avviare da quella pagina.
3. In Actions (Operazioni) scegliere Start (Avvia).

Avvio di un cluster database arrestato utilizzando la AWS CLI

Per avviare un cluster DB arrestato utilizzando l'AWS CLI, chiama il comando start-db-cluster utilizzando il parametro `--db-cluster-identifier` per specificare il cluster DB arrestato che vuoi avviare. Fornisci il nome che hai scelto durante la creazione del cluster database o utilizza un nome di istanza database aggiungendo `-cluster` alla fine del nome.

Example

```
aws neptune start-db-cluster --db-cluster-identifier mydbcluster
```

Avvio di un cluster database arrestato utilizzando l'API di gestione Neptune

Per avviare un cluster database Neptune mediante l'API di gestione Neptune, chiama l'API StartDBCluster utilizzando il parametro `DBCluster` per specificare il cluster database arrestato che vuoi avviare. Fornisci il nome che hai scelto durante la creazione del cluster DB o utilizza un nome di istanza database con `-cluster` aggiunto alla fine del nome.

Svuotamento di un cluster database Amazon Neptune utilizzando l'API di ripristino rapido

La REST API di ripristino rapido Neptune consente di ripristinare un grafo Neptune in modo veloce e semplice, rimuovendo tutti i relativi dati.

Puoi farlo all'interno di un notebook Neptune usando il comando magic di riga %db_reset.

> **Note**
>
> Questa funzionalità è disponibile a partire dalla versione 1.0.4.0 del motore Neptune.

- Nella maggior parte dei casi, un'operazione di ripristino rapido viene completata entro un paio di minuti. La durata può variare leggermente a seconda del carico sul cluster al momento dell'avvio dell'operazione.
- Un'operazione di ripristino rapido non comporta operazioni di I/O aggiuntive.
- Le dimensioni del volume di archiviazione non si riducono dopo un ripristino rapido. Al contrario, l'archiviazione viene riutilizzata quando vengono inseriti nuovi dati. Ciò significa che le dimensioni del volume degli snapshot scattati prima e dopo un'operazione di ripristino rapido saranno le stesse. Anche le dimensioni dei volumi dei cluster ripristinati che utilizzano snapshot creati prima e dopo un'operazione di ripristino rapido saranno le stesse
- Nell'ambito dell'operazione di ripristino, tutte le istanze nel cluster database vengono riavviate.

> **Note**
>
> In casi rari, questi riavvii del server potrebbero anche causare il failover del cluster.

> **Important**
>
> L'utilizzo del ripristino rapido può interrompere l'integrazione del cluster database Neptune con altri servizi. Ad esempio:
>
> - Il ripristino rapido elimina tutti i dati dei flussi dal database e ripristina completamente i flussi. Ciò significa che i consumatori dei flussi potrebbero non funzionare più senza una nuova configurazione.

- Il ripristino rapido rimuove tutti i metadati sulle risorse SageMaker utilizzate da Neptune ML, inclusi i processi e gli endpoint. Continuano a esistere in SageMaker e puoi continuare a utilizzare gli endpoint SageMaker esistenti per le query di inferenza di Neptune ML, ma le API di gestione Neptune ML non funzioneranno più.
- Anche le integrazioni, ad esempio quella della full-text con ElasticSearch, vengono eliminate dal ripristino rapido e devono essere ripristinate manualmente prima di poter essere riutilizzate.

Per eliminare tutti i dati da un cluster database Neptune utilizzando l'API

1. Genera anzitutto un token da utilizzare successivamente per eseguire il ripristino del database. Questo passaggio ha lo scopo di impedire che qualcuno ripristini accidentalmente un database.

   A tale scopo, invia una richiesta `HTTP POST` all'endpoint `/system` sull'istanza writer del cluster database per specificare l'azione `initiateDatabaseReset`.

   Il comando `curl` che utilizza il tipo di contenuto JSON sarebbe:

   ```
   curl -X POST \
     -H 'Content-Type: application/json' \
         https://your_writer_instance_endpoint:8182/system \
     -d '{ "action" : "initiateDatabaseReset" }'
   ```

   Oppure, utilizzando il tipo di contenuto `x-www-form-urlencoded`:

   ```
   curl -X POST \
     -H 'Content-Type: application/x-www-form-urlencoded' \
         https://your_writer_instance_endpoint:8182/system \
     -d 'action=initiateDatabaseReset '
   ```

   La richiesta `initiateDatabaseReset` restituisce il token di ripristino nella sua risposta JSON, ovvero:

   ```
   {
     "status" : "200 OK",
     "payload" : {
       "token" : "new_token_guid"
     }
   ```

```
}
```

Il token rimane valido per un'ora (60 minuti) dopo l'emissione.

Se invii la richiesta a un'istanza reader o all'endpoint dello stato, Neptune genererà un'eccezione `ReadOnlyViolationException`.

Se invii più richieste `initiateDatabaseReset`, solo l'ultimo token generato sarà valido per il secondo passaggio, in cui eseguirai effettivamente il ripristino.

Se il server si riavvia subito dopo la richiesta `initiateDatabaseReset`, il token generato non è più valido e devi inviare una nuova richiesta per ottenerne uno nuovo.

2. Successivamente, devi inviare una richiesta `performDatabaseReset` con il token ricevuto da `initiateDatabaseReset` all'endpoint `/system` sull'istanza writer del cluster database. In questo modo vengono eliminati tutti i dati dal cluster database.

   Il comando `curl` che utilizza il tipo di contenuto JSON è:

   ```
   curl -X POST \
     -H 'Content-Type: application/json' \
         https://your_writer_instance_endpoint:8182/system \
     -d '{
           "action" : "performDatabaseReset",
           "token" : "token_guid"
         }'
   ```

   Oppure, utilizzando il tipo di contenuto `x-www-form-urlencoded`:

   ```
   curl -X POST \
     -H 'Content-Type: application/x-www-form-urlencoded' \
         https://your_writer_instance_endpoint:8182/system \
     -d 'action=performDatabaseReset&token=token_guid'
   ```

   La richiesta restituisce una risposta JSON. Se la richiesta viene accettata, la risposta è:

   ```
   {
     "status" : "200 OK"
   }
   ```

   Se il token che hai inviato non corrisponde a quello emesso, la risposta è simile alla seguente:

```
{
  "code" : "InvalidParameterException",
  "requestId":"token_guid",
  "detailedMessage" : "System command parameter 'token' : 'token_guid' does not
 match database reset token"
}
```

Se la richiesta viene accettata e il ripristino ha inizio, il server si riavvia ed elimina i dati. Non è possibile inviare altre richieste al cluster database durante il ripristino.

Utilizzo dell'API di ripristino rapido con IAM-Auth

Se hai abilitato IAM-Auth nel cluster database, puoi usare awscurl per inviare comandi di ripristino rapido autenticati mediante IAM-Auth:

Utilizzo di awscurl per inviare richieste di ripristino rapido con IAM-Auth

1. Imposta correttamente le variabili di ambiente `AWS_ACCESS_KEY_ID` e `AWS_SECRET_ACCESS_KEY` (e anche `AWS_SECURITY_TOKEN` se stai usando credenziali temporanee).
2. Una richiesta `initiateDatabaseReset` è simile a questa:

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "initiateDatabaseReset" }'
```

3. Una richiesta `performDatabaseReset` è simile a questa:

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "performDatabaseReset" }'
```

Utilizzo del comando magic di riga **`%db_reset`** del workbench Neptune per ripristinare un cluster database

Il workbench Neptune supporta un comando magic di riga `%db_reset` che consente di eseguire un ripristino rapido del database in un notebook Neptune.

Se si richiama il comando magic senza alcun parametro, viene visualizzata una schermata in cui viene chiesto se si desidera eliminare tutti i dati del cluster. C'è anche una casella di controllo che richiedere di essere a conoscenza del fatto che i dati del cluster non saranno più disponibili dopo l'eliminazione. A quel punto, puoi scegliere di procedere con l'eliminazione dei dati o annullare l'operazione.

Un'opzione più rischiosa è richiamare `%db_reset` con l'opzione `--yes` o `-y`, in modo che l'eliminazione venga eseguita senza ulteriori richieste.

Puoi anche eseguire il ripristino in due passaggi, come con REST API:

```
%db_reset --generate-token
```

La risposta è:

```
{
  "status" : "200 OK",
  "payload" : {
    "token" : "new_token_guid"
  }
}
```

Poi esegui:

```
%db_reset --token new_token_guid
```

La risposta è:

```
{
  "status" : "200 OK"
}
```

Codici di errore comuni per operazioni di ripristino rapido

Codice di errore Neptune	Stato HTTP	Message	Esempio
`InvalidParameterException`	400	Il parametro del comando di sistema '*`action`*' ha un valore non supportato '*`XXX`*'	Parametro non valido
`InvalidParameterException`	400	Troppi valori forniti per: *`action`*	Una richiesta di ripristino rapido con più di un'azione inviata con l'intestazione 'Content-type:application/x-www-form-urlencoded'
`InvalidParameterException`	400	Campo 'action' duplicato	Una richiesta di ripristino rapido con più di un'azione inviata con l'intestazione 'Content-Type: application/json'
`MethodNotAllowedException`	400	Percorso errato: /*`bad_endpoint`*	Richiesta inviata a un endpoint errato
`MissingParameterException`	400	Parametri obbligatori mancanti: [action]	Una richiesta di ripristino rapido non contiene il parametro 'action' richiesto
`ReadOnlyViolationException`	400	Le scritture non sono consentite su	È stata inviata una richiesta di ripristin

Codice di errore Neptune	Stato HTTP	Message	Esempio
		un'istanza di replica di lettura	o rapido endpoint di reader o di stato
`AccessDeniedException`	403	Token di autenticazione mancante	Una richiesta di ripristino rapido è stata inviata senza firme corrette a un endpoint di database in cui è abilitato IAM-Auth
`ServerShutdownException`	500	Il ripristino del database è in corso. Riprova a eseguire la query dopo che il cluster diventa disponibile.	Quando inizia il ripristino rapido, le query Gremlin/Sparql esistenti e in entrata non riescono.

Aggiunta di istanze reader Neptune a un cluster database

Nei cluster database Neptune possono esistere un'istanza database primaria e fino a 15 istanze reader Neptune. L'istanza database primaria supporta operazioni di lettura e scrittura ed esegue tutte le modifiche ai dati del volume del cluster. Le istanze reader Neptune si connettono allo stesso volume di archiviazione dell'istanza database primaria e supporta solo operazioni di lettura.

Utilizzare le istanze reader per effettuare l'offload dei carichi di lavoro in lettura dall'istanza database primaria.

Consigliamo di distribuire l'istanza primaria e le istanze reader Neptune nel cluster database su più zone di disponibilità, in modo da migliorare la disponibilità del cluster database.

La sezione seguente descrive come creare un'istanza reader nel cluster database.

Creazione di un'istanza reader Neptune mediante la console

Dopo aver creato l'istanza primaria per il cluster database Neptune, puoi aggiungere altre istanze reader Neptune utilizzando la console Neptune.

Per creare un'istanza reader Neptune mediante la AWS Management Console

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, scegli Database.
3. Scegli il cluster in cui creare l'istanza reader.
4. Scegli Azioni e poi Aggiungi istanza.
5. Nella pagina Crea un'istanza database di replica di lettura, specifica le opzioni per la replica Neptune. La tabella riportata di seguito mostra le impostazioni di una replica di lettura Neptune.

Per questa opzione...	eseguire questa operazione
DB instance class (Classe istanza database)	Seleziona una classe di istanza database che definisca i requisiti di elaborazione e di memoria per la replica Neptune. Per un elenco attuale delle classi di istanza database offerte da Neptune in regioni diverse, consultala pagina dei prezzi Neptune.
Availability zone (Zona di disponibilità)	Specifica una zona di disponibilità. Scegli un'altra zona rispetto a quella dell'istanza database primaria. L'elenco include solo le zone di disponibilità mappate tramite il gruppo di sottoreti del database per il cluster di database.
Encryption (Crittografia)	Abilita o disabilita la crittografia.
Read replica source (Origine replica di lettura)	Seleziona l'identificatore dell'istanza primaria per cui creare una replica Neptune.
DB instance identifier (Identificatore istanze DB)	Inserisci un nome per l'istanza che sia univoco per l'account nella regione selezionata. Puoi scegliere di aggiungere alcune informazioni utili al nome come

Per questa opzione...	eseguire questa operazione
	la zona di disponibilità selezionata, ad esempio `neptune-us-east-1c` .
Database port (Porta del database)	Numero della porta sulla quale il database accetta connessioni.
DB parameter group (Gruppo di parametri database)	Il gruppo di parametri per questa istanza.
Log exports (Esportazioni log)	Scegli i log che desideri pubblicare, se ce ne sono.
Auto Minor Version Upgrade (Aggiornamento automatico versioni secondarie)	Scegli Sì per abilitare la replica Neptune in modo che riceva automaticamente gli aggiornamenti delle versioni secondarie del motore di database Neptune non appena diventano disponibili. L'opzione Auto Minor Version Upgrade (Aggiorna mento automatico versione secondaria) si applica solo agli aggiornamenti secondari. Non si applica alle patch di manutenzione del motore, che vengono sempre applicate automaticamente per mantenere la stabilità del sistema.

6. Seleziona Crea replica di lettura per creare l'istanza di replica Neptune.

Per rimuovere un'istanza reader Neptune da un cluster database, segui le istruzioni in Eliminazione di un'istanza database in Amazon Neptune.

Modifica di un cluster database Neptune mediante la console

Quando modifichi un'istanza database utilizzando la AWS Management Console, puoi scegliere di applicare subito le modifiche selezionando l'opzione Apply Immediately (Applica immediatamente). Se scegli di applicare le modifiche immediatamente, verranno applicate tutte insieme le nuove modifiche e tutte le modifiche in sospeso nella coda.

Se non scegli di applicare le modifiche immediatamente, le modifiche vengono inserite nella coda delle modifiche in sospeso. Durante la finestra di manutenzione successiva, le eventuali modifiche in sospeso incluse nella coda vengono eseguite.

> **Important**
>
> Se una delle modifiche in sospeso richiede tempi di inattività, scegliere di applicarle immediatamente può provocare tempi di inattività imprevisti per l'istanza database in questione. Non sono previsti tempi di inattività per le altre istanze database nel cluster di database.

> **Note**
>
> Quando modifichi un cluster database in Neptune, l'impostazione Applica immediatamente si applica solo alle modifiche alle impostazioni Identificatore cluster database e Autenticazione database IAM. Tutte le altre modifiche vengono applicate immediatamente, indipendentemente dal valore dell'impostazione Apply Immediately (Applica immediatamente).

Per avviare un cluster di database tramite la console

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione scegli Cluster, quindi scegli il cluster da modificare.
3. Scegli Actions (Operazioni) e quindi Modify cluster (Modifica cluster). Viene visualizzata la pagina Modify DB cluster (Modifica cluster database).
4. Modificare le impostazioni desiderate.

> **Note**
>
> Nella console, alcune modifiche a livello di istanza si applicano solo all'istanza database corrente, mentre altre si applicano all'intero cluster di database. Per cambiare un'impostazione che consente di modificare l'intero cluster di database a livello di istanza nella console, segui le istruzioni contenute in Modificare un'istanza database in un cluster di database.

5. Quando tutte le modifiche sono come le desideri, seleziona Continue (Continua) e controlla il riepilogo.
6. Per applicare le modifiche immediatamente, selezionare Apply Immediately (Applica immediatamente).
7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, selezionare Modify cluster (Modifica cluster) per salvare le modifiche.

   Per cambiare le modifiche, scegli Back (Indietro) oppure, per annullare le modifiche, scegli Cancel (Annulla).

Modificare un'istanza database in un cluster di database

Modifica di un'istanza database in un cluster di database utilizzando la console

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, scegli Instances (Istanze) e l'istanza database da modificare.
3. Scegliere Instance actions (Operazioni istanza), quindi Modify (Modifica). Viene visualizzata la pagina Modify DB Instance (Modifica istanza database).
4. Modificare le impostazioni desiderate.

> **Note**
>
> Alcune impostazioni sono valide per l'intero cluster di database e devono essere modificate a livello di cluster. Per modificare queste impostazioni, seguire le istruzioni in Modifica di un cluster database Neptune mediante la console.

Nella AWS Management Console alcune modifiche a livello di istanza si applicano solo all'istanza database corrente, mentre altre si applicano all'intero cluster di database.

5. Quando tutte le modifiche sono come le desideri, seleziona Continue (Continua) e controlla il riepilogo.
6. Per applicare le modifiche immediatamente, selezionare Apply Immediately (Applica immediatamente).
7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, seleziona Modifica istanza database per salvare le modifiche.

   Per cambiare le modifiche, scegli Back (Indietro) oppure, per annullare le modifiche, scegli Cancel (Annulla).

Prestazioni e dimensionamento in Amazon Neptune

Il dimensionamento di cluster e istanze database di Neptune avviene su tre livelli diversi:

- Dimensionamento dello storage
- Dimensionamento delle istanze;
- Dimensionamento della lettura

Dimensionamento dell'archiviazione in Neptune

L'archiviazione Neptune viene dimensionata automaticamente con i dati del volume dei cluster. Man mano che i dati aumentano, l'archiviazione del volume del cluster aumenta fino a 128 TiB in tutte le regioni supportate, ad eccezione della Cina e di GovCloud, in cui è limitato a 64 TiB.

Le dimensioni del volume cluster vengono controllate ogni ora, per stabilire i costi di storage.

I costi per l'archiviazione utilizzata dal database Neptune vengono addebitati in incrementi di GB al mese e gli I/O utilizzati vengono addebitati in incrementi di un milione di richieste. I prezzi sono calcolati in base all'archiviazione e alle operazioni I/O utilizzate dal database Neptune e non è necessario alcun provisioning anticipato.

Per informazioni sui prezzi, consulta la pagina del prodotto di Neptune.

Dimensionamento delle istanze in Neptune

Puoi dimensionare il cluster database Neptune a seconda delle esigenze modificando la classe per ogni istanza database nel cluster database. Neptune supporta diverse classi di istanza database con memoria ottimizzata.

Dimensionamento della lettura in Neptune

Puoi ottenere il dimensionamento della lettura per il cluster database Neptune creando fino a 15 repliche di Neptune nel cluster database. Ogni replica di Neptune restituisce gli stessi dati dal volume del cluster con ritardo di replica minimo, in genere di molto inferiore a 100 millisecondi dopo che l'istanza primaria ha scritto un aggiornamento. Man mano che il traffico di lettura aumenta, puoi creare altre repliche di Neptune ed effettuare direttamente una connessione alle stesse per distribuire il carico di lettura del cluster database. Non è necessario che le repliche di Neptune appartengano alla stessa classe di istanze database dell'istanza primaria.

Per informazioni sull'aggiunta di repliche di Neptune a un cluster database, consulta Aggiunta di istanze reader.

Dimensionamento automatico del numero di repliche in un cluster database Amazon Neptune

È possibile utilizzare il dimensionamento automatico di Neptune per regolare automaticamente il numero di repliche Neptune in un cluster database per soddisfare i requisiti di connettività e carico di lavoro. Il dimensionamento automatico consente al cluster database Neptune di gestire gli aumenti del carico di lavoro e, quando questo cala, rimuove le repliche non necessarie in modo da non pagare per la capacità inutilizzata.

È possibile utilizzare il dimensionamento automatico solo con un cluster database Neptune che ha già un'istanza writer principale e almeno un'istanza di replica di lettura (vedi Cluster e istanze database di Amazon Neptune). Inoltre, tutte le istanze di replica di lettura nel cluster devono essere in stato disponibile. Se una replica di lettura si trova in uno stato non disponibile, il dimensionamento automatico di Neptune non fa nulla finché non sono disponibili tutte le repliche di lettura nel cluster.

Se hai bisogno di creare un nuovo cluster, consulta Creazione di un cluster DB.

Utilizza AWS CLI per definire e applicare una policy di dimensionamento al cluster database. Puoi usare AWS CLI anche per modificare o eliminare la policy di dimensionamento automatico. La policy specifica i seguenti parametri di dimensionamento automatico:

- Il numero minimo e massimo di repliche che deve avere il cluster.
- Un intervallo `ScaleOutCooldown` tra le aggiunte delle repliche e un intervallo `ScaleInCooldown` tra le eliminazioni delle repliche.
- La metrica CloudWatch e il valore di attivazione della metrica per l'aumento o la riduzione del dimensionamento.

La frequenza delle azioni di dimensionamento automatico di Neptune viene ridotta in diversi modi:

- Inizialmente, affinché il dimensionamento automatico aggiunga o elimini una replica, l'allarme `CPUUtilization` per il livello massimo deve essere violato per almeno 3 minuti, mentre quello per il livello minimo deve essere violato per almeno 15 minuti.
- Dopo la prima aggiunta o eliminazione, la frequenza delle successive azioni di dimensionamento automatico di Neptune è limitata dalle impostazioni `ScaleOutCooldown` e `ScaleInCooldown` nella policy di dimensionamento automatico.

Se la metrica CloudWatch in uso raggiunge la soglia massima specificata nella policy, se è trascorso l'intervallo `ScaleOutCooldown` dall'ultima azione di dimensionamento automatico e se il cluster database non ha già il numero massimo di repliche che hai impostato, il dimensionamento automatico di Neptune crea una nuova replica utilizzando lo stesso tipo di istanza di quella principale del cluster database.

Allo stesso modo, se la metrica raggiunge la soglia minima specificata, se è trascorso l'intervallo `ScaleInCooldown` dall'ultima azione di dimensionamento automatico e se il cluster database ha più repliche del numero minimo specificato, il dimensionamento automatico di Neptune elimina una delle repliche.

> **Note**
>
> Il dimensionamento automatico di Neptune rimuove solo le repliche che ha creato. Non rimuove le repliche preesistenti.

Utilizzando il parametro neptune_autoscaling_config del cluster database è possibile anche specificare il tipo di istanza delle nuove repliche di lettura create dal dimensionamento automatico di Neptune, le finestre di manutenzione per tali repliche di lettura e i tag da associare a ciascuna delle nuove repliche di lettura. Queste impostazioni di configurazione vengono fornite in una stringa JSON come valore del parametro `neptune_autoscaling_config`, come descritto di seguito:

```
"{
  \"tags\": [
    { \"key\" : \"reader tag-0 key\", \"value\" : \"reader tag-0 value\" },
    { \"key\" : \"reader tag-1 key\", \"value\" : \"reader tag-1 value\" },
  ],
  \"maintenanceWindow\" : \"wed:12:03-wed:12:33\",
  \"dbInstanceClass\" : \"db.r5.xlarge\"
}"
```

Nota che tutte le virgolette all'interno della stringa JSON devono essere precedute dal carattere di escape della barra rovesciata (\). Come sempre, tutti gli spazi nella stringa sono facoltativi.

Tutte le tre impostazioni di configurazione non specificate nel parametro `neptune_autoscaling_config` vengono copiate dalla configurazione dell'istanza writer primaria del cluster database.

Quando il dimensionamento automatico aggiunge una nuova istanza di replica di lettura, aggiunge all'ID dell'istanza database il prefisso `autoscaled-reader` (ad esempio `autoscaled-reader-7r7t7z3lbd-20210828`). Aggiunge inoltre un tag a ogni replica di lettura che crea con la chiave `autoscaled-reader` e il valore `TRUE`. È possibile visualizzarlo nella scheda Tag della pagina dei dettagli dell'istanza database nella AWS Management Console.

```
"key" : "autoscaled-reader",  "value" : "TRUE"
```

Il livello di promozione di tutte le istanze di replica di lettura create dal dimensionamento automatico è la priorità più bassa, ovvero 15 per impostazione predefinita. Ciò significa che, durante un failover una replica con una priorità più alta, ad esempio una creata manualmente, sarà promossa per prima. Per informazioni, consultare Tolleranza ai guasti di un cluster database Neptune..

Il dimensionamento automatico di Neptune è implementato utilizzando Application Auto Scaling con una policy di dimensionamento di monitoraggio dei target che utilizza `CPUUtilization` come metrica CloudWatch predefinita.

Utilizzo del dimensionamento automatico in un cluster database serverless di Neptune

Neptune Serverless risponde molto più rapidamente del dimensionamento automatico di Neptune quando la domanda supera la capacità di un'istanza e aumenta la scalabilità dell'istanza, anziché aggiungerne un'altra. Mentre il dimensionamento automatico è progettato per far fronte ad aumenti o diminuzioni relativamente stabili del carico di lavoro, Serverless è ottimo per gestire picchi e variazioni impreviste della domanda.

Tenendo presenti questi punti di forza, puoi combinare il dimensionamento automatico e Serverless per creare un'infrastruttura flessibile in grado di gestire le variazioni dei carichi di lavoro in modo efficiente, oltre a soddisfare la domanda riducendo al minimo i costi.

Affinché il dimensionamento automatico possa funzionare in modo efficace con Serverless, è importante configurare l'impostazione `maxNCU` del cluster serverless su un livello sufficientemente alto da far fronte a picchi e brevi variazioni della domanda. In caso contrario, le modifiche transitorie non attivano il dimensionamento serverless, il che potrebbe far sì che il dimensionamento automatico crei molte istanze aggiuntive non necessarie. Se il valore di `maxNCU` è impostato su un livello sufficientemente alto, il dimensionamento serverless può gestire tali modifiche più velocemente e con meno costi.

Come abilitare il dimensionamento automatico per Amazon Neptune

Il dimensionamento automatico può essere abilitato solo per un cluster database Neptune utilizzando AWS CLI. Non è possibile abilitare il dimensionamento automatico utilizzando la AWS Management Console.

Inoltre, il dimensionamento automatico non è supportato nelle seguenti regioni Amazon:

- Africa (Città del Capo): `af-south-1`
- Medio Oriente (Emirati Arabi Uniti): `me-central-1`
- AWS GovCloud (US-East): `us-gov-east-1`
- AWS GovCloud (US-West): `us-gov-west-1`

L'abilitazione del dimensionamento automatico per un cluster database Neptune prevede tre passaggi:

1. Registrazione del cluster database tramite Application Auto Scaling

Il primo passaggio per abilitare il dimensionamento automatico per un cluster database Neptune prevede la registrazione del cluster con Application Auto Scaling utilizzando AWS CLI o uno degli SDK di Application Auto Scaling. Il cluster deve avere già un'istanza primaria e almeno un'istanza di replica di lettura:

Ad esempio, per registrare un cluster da dimensionare automaticamente con altre repliche (da 1 a 8), è possibile utilizzare il comando AWS CLI `register-scalable-target`, come descritto di seguito:

```
aws application-autoscaling register-scalable-target \
  --service-namespace neptune \
  --resource-id cluster:(your DB cluster name) \
  --scalable-dimension neptune:cluster:ReadReplicaCount \
  --min-capacity 1 \
  --max-capacity 8
```

Equivale all'utilizzo dell'operazione `RegisterScalableTarget` dell'API Application Auto Scaling.

Il comando AWS CLI `register-scalable-target` accetta i parametri seguenti:

- **`service-namespace`**: impostato su `neptune`.

Questo parametro equivale al parametro `ServiceNamespace` dell'API Application Auto Scaling.

- **`resource-id`**: impostalo come identificatore di risorsa per il cluster database Neptune. Il tipo di risorsa è `cluster`, seguito da due punti (':') e poi dal nome del cluster database.

  Questo parametro equivale al parametro `ResourceID` dell'API Application Auto Scaling.

- **`scalable-dimension`**: la dimensione scalabile in questo caso è il numero di istanze di replica nel cluster database, quindi occorre impostarlo su `neptune:cluster:ReadReplicaCount`.

  Questo parametro equivale al parametro `ScalableDimension` dell'API Application Auto Scaling.

- **`min-capacity`**: il numero minimo di istanze database di replica di lettura da gestire con Application Auto Scaling. Questo valore deve essere impostato su un intervallo compreso tra 0 e 15 ed essere uguale o minore rispetto al valore specificato per il numero massimo di repliche Neptune in `max-capacity`. Affinché il dimensionamento automatico funzioni, il cluster database deve avere almeno un'istanza di lettura.

  Questo parametro equivale al parametro `MinCapacity` dell'API Application Auto Scaling.

- **`max-capacity`**: il numero massimo di istanze database di replica di lettura nel cluster database, incluse istanze preesistenti e quelle nuove gestite da Application Auto Scaling. Questo valore deve essere impostato su un intervallo compreso tra 0 e 15 ed essere uguale o minore rispetto al valore specificato per il numero minimo di repliche Neptune in `min-capacity`.

  Il parametro `max-capacity` AWS CLI equivale al parametro `MaxCapacity` nell'API Application Auto Scaling.

Quando registri il cluster database, Application Auto Scaling crea un ruolo `AWSServiceRoleForApplicationAutoScaling_NeptuneCluster` collegato ai servizi. Per ulteriori informazioni, consulta Ruoli collegati ai servizi per Application Auto Scaling nella Guida per l'utente di Application Auto Scaling.

2. Definisci una policy di dimensionamento automatico da utilizzare con il cluster database

Una policy di dimensionamento per il monitoraggio dei target è definita come un oggetto di testo JSON che può essere salvato anche in un file di testo. Per Neptune, questa policy attualmente può utilizzare solo `CPUUtilization` come metrica CloudWatch di Neptune predefinita denominata `NeptuneReaderAverageCPUUtilization`.

Di seguito è riportato un esempio di policy di configurazione del monitoraggio dei target per Neptune:

```
{
  "PredefinedMetricSpecification": { "PredefinedMetricType":
 "NeptuneReaderAverageCPUUtilization" },
  "TargetValue": 60.0,
  "ScaleOutCooldown" : 600,
  "ScaleInCooldown" : 600
}
```

L'elemento **`TargetValue`** qui contiene la percentuale di utilizzo della CPU al di sopra della quale il dimensionamento automatico impiega la scalabilità orizzontale (ovvero aggiunge altre repliche) e al di sotto della quale esegue il ridimensionamento (ovvero elimina le repliche). In questo caso, la percentuale target che attiva la scalabilità è pari al `60.0`%.

L'elemento **`ScaleInCooldown`** specifica la quantità di tempo, in secondi, tra il completamento di un'attività di ridimensionamento e l'inizio di un'altra attività dello stesso tipo. Il valore predefinito è 300 secondi. Qui, il valore è 600 e specifica che devono trascorrere almeno dieci minuti tra il completamento dell'eliminazione di una replica e l'inizio di un'altra.

L'elemento **`ScaleOutCooldown`** specifica la quantità di tempo, in secondi, tra il completamento di un'attività di impiego della scalabilità orizzontale e l'inizio di un'altra attività dello stesso tipo. Il valore predefinito è 300 secondi. Qui, il valore è 600 e specifica che devono trascorrere almeno dieci minuti tra il completamento dell'aggiunta di una replica e l'inizio di un'altra.

L'elemento **`DisableScaleIn`** è un valore booleano che, se presente e impostato su `true`, disabilita completamente il ridimensionamento, nel senso che il dimensionamento automatico può aggiungere repliche, ma non ne rimuoverà mai nessuna. Per impostazione predefinita, il ridimensionamento è abilitato e `DisableScaleIn` è `false`.

Dopo la registrazione del cluster database Neptune con Application Auto Scaling e la definizione di una policy di dimensionamento JSON in un file di testo, applica la policy di dimensionamento al cluster database registrato. Puoi eseguire questa operazione utilizzando il comando AWS CLI [put-scaling-policy] con i seguenti parametri:

```
aws application-autoscaling put-scaling-policy \
  --policy-name (name of the scaling policy) \
  --policy-type TargetTrackingScaling \
  --resource-id cluster:(name of your Neptune DB cluster) \
```

```
  --service-namespace neptune \
  --scalable-dimension neptune:cluster:ReadReplicaCount \
  --target-tracking-scaling-policy-configuration file://(path to the JSON configuration
file)
```

Dopo aver applicato la policy, il dimensionamento automatico è abilitato sul cluster database.

Puoi anche utilizzare il comando AWS CLI `put-scaling-policy` per aggiornare una policy di dimensionamento automatico esistente.

Vedi anche PutScalingPolicy nella Guida di riferimento all'API Application AutoScaling.

Rimozione del dimensionamento automatico da un cluster database Neptune

Per rimuovere il dimensionamento automatico da un cluster database Neptune, usa i comandi AWS CLI delete-scaling-policy e deregister-scalable-target.

Gestione del cluster di database Amazon Neptune

Neptune esegue periodicamente la manutenzione di tutte le risorse che utilizza, incluso:

- Sostituzione dell'hardware sottostante, in base alle esigenze. Ciò avviene in background, senza che l'utente esegua alcuna azione, e generalmente non influenza le operazioni.
- Aggiornamento del sistema operativo sottostante. Gli aggiornamenti del sistema operativo delle istanze del cluster di database vengono effettuati per migliorare le prestazioni e la sicurezza, pertanto è in genere consigliabile completarli il prima possibile. In genere, gli aggiornamenti richiedono circa 10 minuti. Gli aggiornamenti del sistema operativo non modificano la versione del motore database o la classe istanza database di un'istanza database.

  In genere, è preferibile aggiornare prima le istanze reader in un cluster di database, quindi l'istanza writer. L'aggiornamento contemporaneo delle istanze reader e writer può causare tempi di inattività in caso di failover. Tieni presente che il backup delle istanze database non viene eseguito automaticamente prima di un aggiornamento del sistema operativo, pertanto assicurati di eseguire backup manuali prima di applicare un aggiornamento del sistema operativo.
- Aggiornamento del motore di database Neptune. Neptune rilascia regolarmente una serie di aggiornamenti del motore per introdurre nuove funzionalità e miglioramenti e per correggere bug.

Numeri di versione del motore

Numerazione delle versioni prima del rilascio del motore 1.3.0.0

Prima di novembre 2019, Neptune supportava solo una versione del motore alla volta e i numeri di versione del motore avevano tutti il formato `1.0.1.0.200`*`<xxx>`*, dove `xxx` corrispondeva al numero di patch. Tutte le nuove versioni del motore sono state tutte rilasciate come patch alle versioni precedenti.

Nel novembre 2019, Neptune ha iniziato a supportare più versioni, consentendo ai clienti un migliore controllo sui rispettivi percorsi di aggiornamento. Di conseguenza, la numerazione dei rilasci del motore è cambiata.

A partire da novembre 2019 fino al rilascio del motore 1.3.0.0, i numeri di versione del motore sono composti da 5 parti. Utilizza il numero di versione `1.0.2.0.R2` come esempio:

- La prima parte è sempre 1.
- La seconda parte, `0` in `1.0.2.0.R2`, è il numero della versione principale del database.

- La terza e la quarta parte, `2.0` in `1.0.2.0.R2`, sono entrambe numeri di versione secondari.
- La quinta parte (`R2` in `1.0.2.0.R2`) è il numero della patch.

La maggior parte degli aggiornamenti erano aggiornamenti della patch e la distinzione tra aggiornamenti della patch e aggiornamenti della versione secondaria non era sempre chiara.

Numerazione delle versioni a partire dal rilascio del motore 1.3.0.0

A partire dal rilascio del motore 1.3.0.0, Neptune ha cambiato il modo in cui gli aggiornamenti del motore vengono numerati e gestiti.

I numeri di versione del motore sono ora composti da quattro parti, ciascuna delle quali corrisponde a un tipo di rilascio, come segue:

versione-prodotto.*versione-principale*.*versione-secondaria*.*versione-patch*

Le modifiche non irreversibili, tipo quelle che erano state rilasciate in precedenza come patch, vengono ora rilasciate come versioni secondarie che è possibile gestire utilizzando l'impostazione dell'istanza `AutoMinorVersionUpgrade`.

Ciò significa che, se lo si desidera, è possibile ricevere una notifica ogni volta che viene rilasciata una nuova versione secondaria, sottoscrivendo l'abbonamento all'evento `RDS-EVENT-0156` (consulta Sottoscrizione alle notifiche eventi Neptune).

I rilasci delle patch vengono ora riservati per correzioni mirate urgenti e sono numerati utilizzando l'ultima parte del numero di versione (`*.*.*.1`, `*.*.*.2` e così via).

Tipi diversi di rilasci del motore in Amazon Neptune

I quattro tipi di rilascio del motore che corrispondono alle quattro parti di un numero di versione del motore sono i seguenti:

- Versione del prodotto: cambia solo se il prodotto subisce modifiche radicali, fondamentali in termini di funzionalità o interfaccia. La versione corrente del prodotto Neptune è 1.
- Versione principale: le versioni principali introducono nuove importanti funzionalità e modifiche speciali e generalmente hanno una durata utile di almeno due anni.
- Versione secondaria: le versioni secondarie possono contenere nuove funzionalità, miglioramenti e correzioni di bug, ma non contengono modifiche speciali. Puoi scegliere se applicarle o meno automaticamente durante la successiva finestra di manutenzione e puoi anche scegliere di ricevere una notifica ogni volta che ne viene rilasciata una.

- Versione della patch: le versioni delle patch vengono rilasciate solo per risolvere correzioni di bug urgenti o aggiornamenti di sicurezza critici. Raramente contengono modifiche speciali e vengono applicate automaticamente durante la finestra di manutenzione successiva al loro rilascio.

Aggiornamenti della versione principale di Amazon Neptune

Un aggiornamento della versione principale in genere introduce una o più nuove funzionalità importanti e spesso contiene modifiche speciali. Di solito ha una durata del supporto di circa due anni. Le versioni principali di Neptune sono elencate in Rilasci del motore, insieme alla data di rilascio e alla durata stimata.

Gli aggiornamenti della versione principale sono del tutto facoltativi finché la versione principale che si sta utilizzando non raggiunge il suo fine vita. Se si sceglie di eseguire l'aggiornamento a una nuova versione principale, occorre installare manualmente la nuova versione utilizzando la AWS CLI o la console Neptune come descritto in Aggiornamenti di una versione principale.

Se la versione principale che si sta utilizzando raggiunge il suo fine vita, tuttavia, si riceverà una notifica con la richiesta di eseguire l'aggiornamento a una versione principale più recente. Quindi, se non si effettua l'aggiornamento entro un periodo di tolleranza dopo la notifica, un aggiornamento alla versione principale più recente viene pianificato automaticamente durante la finestra di manutenzione successiva. Per ulteriori informazioni, consulta Durata della versione del motore.

Aggiornamenti della versione secondaria di Amazon Neptune

La maggior parte degli aggiornamenti del motore Neptune sono aggiornamenti della versione secondaria. Si verificano abbastanza frequentemente e non contengono modifiche speciali.

Se il campo `AutoMinorVersionUpgrade` è impostato su `true` nell'istanza writer (primaria) del cluster di database, gli aggiornamenti della versione secondaria vengono applicati automaticamente a tutte le istanze del cluster di database durante la finestra di manutenzione successiva dopo che vengono rilasciati.

Se il campo `AutoMinorVersionUpgrade` è stato impostato su `false` nell'istanza writer del cluster di database, vengono applicati solo se installi in maniera esplicita.

> **Note**
>
> Gli aggiornamenti delle versioni secondarie sono autonomi (non dipendono da aggiornamenti precedenti della versione secondaria alla stessa versione principale) e cumulativi

(contengono tutte le funzionalità e le correzioni introdotte negli aggiornamenti precedenti della versione secondaria). Ciò significa che è possibile installare qualsiasi aggiornamento della versione secondaria specificato a prescindere che siano stati installati quelli precedenti.

È facile tenere traccia dei rilasci della versione secondaria sottoscrivendo un abbonamento all'evento `RDS-EVENT-0156` (consulta Sottoscrizione alle notifiche eventi Neptune). Ogni volta che viene rilasciata una nuova versione secondaria, si riceverà una notifica.

Inoltre, a prescindere che si sottoscriva un abbonamento alle notifiche, è sempre possibile controllare quali aggiornamenti sono in sospeso.

Aggiornamenti della versione della patch di Amazon Neptune

In caso di problemi di sicurezza o altri difetti gravi che influenzano l'affidabilità dell'istanza, Neptune distribuisce patch obbligatorie. Queste vengono applicate a tutte le istanze del cluster di database durante la successiva finestra di manutenzione senza alcun intervento da parte dell'utente.

Un rilascio patch viene implementato solo quando i rischi di non implementarlo superano i rischi e i tempi di inattività associati alla sua implementazione. I rilasci patch sono poco frequenti (in genere una volta ogni pochi mesi) e talvolta richiedono una parte considerevole della finestra di manutenzione.

Pianificazione della durata della versione principale del motore Amazon Neptune

Le versioni del motore Neptune raggiungono quasi sempre la fine del loro ciclo di vita al termine di un trimestre di calendario. Le eccezioni si verificano solo quando sorgono importanti problemi di sicurezza o disponibilità.

Quando una versione del motore raggiunge la fine del ciclo di vita, ti verrà richiesto di aggiornare il database Neptune a una versione più recente.

In generale, le versioni del motore Neptune continueranno a essere disponibili come segue:

- Versioni secondarie del motore: le versioni secondarie del motore rimangono disponibili per almeno sei mesi dopo il rilascio.
- Versioni principali del motore: le versioni principali del motore rimangono disponibili per almeno 12 mesi dopo il rilascio.

Almeno tre mesi prima che una versione del motore raggiunga la fine del ciclo di vita, AWS invierà una notifica e-mail automatica all'indirizzo e-mail associato al tuo account AWS e pubblicherà lo stesso messaggio sul Dashboard AWS Health. In questo modo avrai tempo per pianificare e prepararti per l'aggiornamento.

Quando una versione del motore raggiunge la fine del ciclo di vita, non sarà più possibile creare nuovi cluster o istanze utilizzando tale versione e la scalabilità automatica non potrà più creare istanze utilizzando tale versione.

Una versione del motore che raggiunge effettivamente la fine del ciclo di vita verrà aggiornata automaticamente durante una finestra di manutenzione. Il messaggio che ti verrà inviato tre mesi prima della fine del ciclo di vita della versione del motore conterrà informazioni dettagliate sull'aggiornamento automatico, inclusa la versione implementata con l'aggiornamento automatico, l'impatto sui tuoi cluster database e le azioni consigliate.

Important

È tua responsabilità mantenere aggiornate le versioni del motore di database. AWS consiglia a tutti i clienti di aggiornare i propri database alla versione più recente del motore per usufruire delle più recenti misure di sicurezza, privacy e disponibilità. Se utilizzi il database su un motore o un software non supportato oltre la data di obsolescenza ("motore legacy"), aumenterà la probabilità del verificarsi di problemi di sicurezza, privacy e operativi, inclusi tempi di inattività.
Il funzionamento del database su qualsiasi motore è soggetto all'accordo che disciplina l'utilizzo dei Servizi AWS. I motori legacy non sono disponibili a livello generale. AWS non fornisce più il supporto per il motore legacy e AWS potrebbe imporre limiti all'accesso o all'uso di qualsiasi motore legacy in qualunque momento, se AWS determina che il motore legacy rappresenta un rischio per la sicurezza o la responsabilità o un rischio di danno per i Servizi AWS, le sue affiliate o terze parti. La decisione dell'utente di continuare a utilizzare il proprio contenuto in un motore legacy potrebbe comportare l'indisponibilità, il danneggiamento o l'irrecuperabilità del contenuto dell'utente. I database in esecuzione su un motore Legacy sono soggetti alle eccezioni dell'Accordo sul livello di servizio (SLA).
I DATABASE E IL SOFTWARE CORRELATO IN ESECUZIONE SU UN MOTORE LEGACY CONTENGONO BUG, ERRORI, DIFETTI E/O COMPONENTI DANNOSI. DI CONSEGUENZA, E NONOSTANTE QUALSIASI DISPOSIZIONE CONTRARIA CONTENUTA NELL'ACCORDO O NEI TERMINI DEL SERVIZIO, AWS FORNISCE IL MOTORE LEGACY "COSÌ COM'È".

Gestione degli aggiornamenti del motore per il cluster di database Neptune

> **Note**
>
> Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede un riavvio del database per tutte le istanze, quindi è necessario prevedere un tempo di inattività compreso tra 20 o 30 secondi e alcuni minuti, al termine del quale sarà possibile riprendere l'utilizzo del cluster database. In rari casi, potrebbe essere necessario un failover Multi-AZ affinché un aggiornamento di manutenzione su un'istanza venga completato.
>
> Per gli aggiornamenti della versione principale che possono richiedere più tempo per essere applicati, puoi utilizzare una strategia di implementazione blu/verde per ridurre al minimo i tempi di inattività.

Determinazione della versione del motore attualmente in uso

Puoi utilizzare il comando AWS CLI `get-engine-status` per verificare quale versione di rilascio del motore è attualmente utilizzata dal cluster di database:

```
aws neptunedata get-engine-status
```

L'output JSON include un campo `"dbEngineVersion"` simile a:

```
  "dbEngineVersion": "1.3.0.0",
```

Verifica degli aggiornamenti in sospeso e disponibili

Puoi verificare la presenza di aggiornamenti in sospeso al cluster di database utilizzando la console Neptune. Seleziona Database nella colonna di sinistra, quindi seleziona il cluster di database nel riquadro Database. Gli aggiornamenti in sospeso sono elencati nella colonna Manutenzione. Se si seleziona Azioni e quindi Manutenzione, sono disponibili tre soluzioni:

- Aggiornare ora.
- Aggiornare alla finestra successiva.
- Rinviare l'aggiornamento.

Puoi elencare gli aggiornamenti del motore in sospeso utilizzando la AWS CLI come segue:

```
aws neptune describe-pending-maintenance-actions \
  --resource-identifier (ARN of your DB cluster)
  --region (your region) \
  --engine neptune
```

Puoi anche elencare gli aggiornamenti del motore disponibili usando la AWS CLI come segue:

```
aws neptune describe-db-engine-versions \
  --region (your region) \
  --engine neptune
```

L'elenco delle release del motore disponibili include solo quelle che dispongono di un numero di versione superiore a quello corrente e per cui è definito un percorso di aggiornamento.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Un aggiornamento secondario può introdurre nuove funzionalità o comportamenti che possono influenzare il codice senza alcuna modifica speciale.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di eseguire l'aggiornamento del cluster di database di produzione è utilizzare la soluzione di implementazione blu/verde Neptune. In questo modo è possibile eseguire applicazioni e query sulla nuova versione senza influenzare il cluster di database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

Finestra di manutenzione Neptune

La finestra di manutenzione settimanale è un periodo di 30 minuti durante il quale vengono applicati aggiornamenti del motore pianificati e altre modifiche di sistema. La maggior parte degli eventi di manutenzione viene completata durante la finestra di manutenzione di 30 minuti, sebbene eventi di manutenzione di dimensioni maggiori potrebbe richiedere a volte più tempo per il completamento.

Ogni cluster di database dispone di una finestra di manutenzione settimanale di 30 minuti. Se non si specifica un orario preferito per la finestra di manutenzione quando si crea il cluster di database, Neptune seleziona in modo casuale un giorno della settimana e quindi assegna in modo casuale un periodo di 30 minuti al suo interno da un blocco temporale di 8 ore che varia in base alla regione.

Di seguito sono riportati, ad esempio, i blocchi temporali di 8 ore per le finestre di manutenzione utilizzate in diverse regioni AWS:

Region	Periodo di tempo
Stati Uniti occidentali (Oregon)	06:00 - 14:00 UTC
Regione Stati Uniti occidentali (California settentrionale)	06:00 - 14:00 UTC
Stati Uniti orientali (Ohio)	03:00 - 11:00 UTC
Europa (Irlanda)	22:00 - 06:00 UTC

La finestra di manutenzione determina l'ora di inizio delle operazioni in sospeso e la maggior parte delle operazioni di manutenzione viene completata all'interno della finestra, ma attività di manutenzione di dimensioni maggiori possono continuare oltre l'ora di fine della finestra.

Spostamento della finestra di manutenzione del cluster di database

Idealmente, la finestra di manutenzione deve essere eseguita nel momento di utilizzo più basso del cluster. Se ciò non si verifica, è possibile spostarla in un orario migliore, come riportato di seguito:

Per modificare a finestra di manutenzione del cluster di database

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, scegli Database.
3. Scegliere il cluster database per cui si desidera modificare la finestra di manutenzione.
4. Scegliere Modify (Modifica).
5. Scegli Mostra altro nella parte inferiore della pagina Modifica cluster.
6. Nella sezione Finestra di manutenzione preferita, imposta il giorno, l'ora e la durata della finestra di manutenzione come preferisci.
7. Seleziona Successivo.

   Nella pagina di conferma esaminare le modifiche.
8. Per applicare immediatamente le modifiche alla finestra di manutenzione, selezionare Apply immediately (Applica immediatamente).
9. Scegli Invia per applicare le modifiche.

   Per cambiare le modifiche, scegli Precedente oppure, per annullare le modifiche, scegli Annulla.

Utilizzo di **`AutoMinorVersionUpgrade`** per controllare gli aggiornamenti automatici della versione secondaria

> **Important**
>
> `AutoMinorVersionUpgrade` è efficace solo per gli aggiornamenti della versione secondaria successivi alla release del motore 1.3.0.0.

Se il campo `AutoMinorVersionUpgrade` è impostato su `true` nell'istanza writer (primaria) del cluster di database, gli aggiornamenti della versione secondaria vengono applicati automaticamente a tutte le istanze del cluster di database durante la finestra di manutenzione successiva dopo che vengono rilasciati.

Se il campo `AutoMinorVersionUpgrade` è stato impostato su `false` nell'istanza writer del cluster di database, vengono applicati solo se installi in maniera esplicita.

> **Note**
>
> Le release delle patch (`*.*.*.1`, `*.*.*.2` e così via) vengono sempre installate automaticamente durante la successiva finestra di manutenzione, a prescindere dall'impostazione del parametro `AutoMinorVersionUpgrade`.

Puoi impostare `AutoMinorVersionUpgrade` utilizzando la AWS Management Console come indicato di seguito:

Per impostare **`AutoMinorVersionUpgrade`** mediante la console Neptune

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegli l'istanza (writer) primaria del cluster di database per cui desideri impostare `AutoMinorVersionUpgrade`.
4. Scegli Modifica.
5. Scegli Mostra altro nella parte inferiore della pagina Modifica cluster.
6. Nella parte inferiore della pagina espansa, scegli Attiva l'aggiornamento automatico della versione secondaria o Disattiva l'aggiornamento automatico della versione secondaria.
7. Seleziona Successivo.

   Nella pagina di conferma esaminare le modifiche.
8. Per applicare le modifiche all'aggiornamento automatico della versione secondaria, seleziona Applica immediatamente.
9. Scegli Invia per applicare le modifiche.

   Per cambiare le modifiche, scegli Precedente oppure, per annullare le modifiche, scegli Annulla.

Puoi anche usare la AWS CLI per impostare il campo `AutoMinorVersionUpgrade`. Ad esempio, per impostarlo su `true`, puoi utilizzare un comando come il seguente:

```
aws neptune modify-db-instance \
```

```
  --db-instance-identifier (the ID of your cluster's writer instance) \
  --auto-minor-version-upgrade \
  --apply-immediately
```

Allo stesso modo, per impostarlo su `false`, utilizza un comando come il seguente:

```
aws neptune modify-db-instance \
  --db-instance-identifier (the ID of your cluster's writer instance) \
  --no-auto-minor-version-upgrade \
  --apply-immediately
```

Installazione manuale degli aggiornamenti del motore Neptune

Installazione di un aggiornamento della versione principale del motore

Le release del motore principali devono essere sempre installate manualmente. Per ridurre al minimo i tempi di inattività e concedere tempo sufficiente per i test e la convalida, il modo migliore per installare una nuova versione principale consiste generalmente nell'utilizzare la soluzione di implementazione blu/verde Neptune.

In alcuni casi puoi anche utilizzare il modello AWS CloudFormation utilizzato per creare il cluster di database per installare un aggiornamento della versione principale (consulta Utilizzo di un AWS CloudFormation modello per aggiornare la versione del motore del cluster Neptune DB).

Se desideri installare immediatamente un aggiornamento della versione principale, puoi utilizzare un comando CLI come il seguente:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (identifier for your neptune cluster) \
  --engine neptune \
  --engine-version (the new engine version) \
  --apply-immediately
```

Assicurati di specificare la versione del motore che desideri aggiornare. In caso contrario, il motore potrebbe essere aggiornato a una versione che non è la più recente o quella che ti aspetti.

Invece di `--apply-immediately`, puoi specificare `--no-apply-immediately`.

Se il cluster utilizza un gruppo di parametri del cluster personalizzato, assicurati di specificarlo utilizzando questo parametro:

```
  --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Analogamente, se eventuali istanze del cluster utilizzano un gruppo di parametri database personalizzato, assicurati di specificarlo utilizzando questo parametro:

```
  ---db-instance-parameter-group-name (name of the custom instance parameter group)
```

Installazione di un aggiornamento della versione minore del motore utilizzando la AWS Management Console

Per eseguire un aggiornamento della versione minore utilizzando la console Neptune

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, scegli Database, quindi seleziona il cluster di database che desideri modificare.
3. Scegli Modifica.
4. In Specifiche dell'istanza, scegli la nuova versione per cui eseguire l'aggiornamento.
5. Seleziona Successivo.
6. Se desideri applicare le modifiche immediatamente, scegli Applica immediatamente.
7. Scegli Invia per aggiornare il cluster di database.

Installazione di un aggiornamento della versione minore del motore utilizzando la AWS CLI

Puoi utilizzare un comando simile al seguente per eseguire un aggiornamento della versione secondaria senza attendere la finestra di manutenzione successiva:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (your-neptune-cluster) \
  --engine-version (new-engine-version) \
  --apply-immediately
```

Se esegui l'aggiornamento manualmente utilizzando la AWS CLI, assicurati di includere la versione del motore per cui eseguire l'aggiornamento. In caso contrario, il motore potrebbe essere aggiornato a una versione che non è la più recente o quella che ti aspetti.

Aggiornamento alla versione del motore 1.2.0.0 o successiva da una versione precedente alla versione 1.2.0.0

Nella release del motore 1.2.0.0 sono state introdotte numerose modifiche significative che rendono l'operazione di aggiornamento da una versione precedente più complessa del solito:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento potrebbe raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Utilizzo di un AWS CloudFormation modello per aggiornare la versione del motore del cluster Neptune DB

Puoi riutilizzare il modello AWS CloudFormation Neptune che hai usato per creare il tuo Neptune DB Cluster per aggiornarne la versione del motore.

Gli aggiornamenti del motore di Neptune possono riguardare la versione principale o secondaria. L'utilizzo di un AWS CloudFormation modello può aiutare con gli aggiornamenti delle versioni principali, che spesso contengono modifiche significative. Poiché gli aggiornamenti a una versione principale possono contenere modifiche al database non compatibili con le versioni precedenti delle applicazioni esistenti, potrebbe essere necessario apportare modifiche alle applicazioni durante l'aggiornamento. Prima di procedere a un aggiornamento, è consigliabile sempre eseguire un test e creare uno snapshot manuale del cluster database.

Tieni presente che devi eseguire un aggiornamento del motore separato per ogni versione principale. Non puoi saltare una versione principale ed eseguire l'aggiornamento direttamente alla versione principale seguente.

Prima del 17 maggio 2023, se utilizzavi lo stack AWS CloudFormation Neptune per aggiornare la versione del tuo motore, creava semplicemente un nuovo cluster DB vuoto al posto di quello attuale. A partire dal 17 maggio 2023, tuttavia, lo stack AWS CloudFormation Neptune ora supporta gli aggiornamenti del motore in loco che preservano i dati esistenti.

Per un aggiornamento della versione principale, nel modello devono essere impostate le seguenti proprietà in `DBCluster`:

- `DBClusterParameterGroup` (personalizzato o predefinito)
- `DBInstanceParameterGroupName`
- `EngineVersion`

Allo stesso modo, per le istanze database collegate al cluster database è necessario impostare:

- `DBParameterGroup` (personalizzato/predefinito)

Assicurati che nel modello siano definiti tutti i gruppi di parametri, sia predefiniti che personalizzati.

Nel caso di un gruppo di parametri personalizzato, assicurati che la famiglia del gruppo di parametri personalizzati esistente sia compatibile con la nuova versione del motore. Le versioni del motore

precedenti alla 1.2.0.0 utilizzavano una famiglia di gruppi di parametri `neptune1`, mentre le versioni del motore dalla 1.2.0.0 in poi richiedono una famiglia di gruppi di parametri `neptune1.2`. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.

Per gli aggiornamenti della versione principale del motore, specifica un gruppo di parametri con la famiglia appropriata nel campo `DBCluster` `DBInstanceParameterGroupName`.

Un gruppo di parametri predefinito deve essere aggiornato a uno compatibile con la nuova versione del motore.

Tieni presente che Neptune riavvia automaticamente le istanze database dopo un aggiornamento del motore.

Argomenti

- Esempio: aggiornamento della versione secondaria del motore da 1.2.0.1 a 1.2.0.2
- Esempio: aggiornamento della versione principale da 1.1.1.0 a 1.2.0.2 con gruppi di parametri predefiniti
- Esempio: aggiornamento della versione principale da 1.1.1.0 a 1.2.0.2 con gruppi di parametri predefiniti
- Esempio: aggiornamento della versione principale da 1.1.1.0 a 1.2.0.2 con una combinazione di gruppi di parametri predefiniti e personalizzati

Esempio: aggiornamento della versione secondaria del motore da 1.2.0.1 a 1.2.0.2

Individua il cluster database che desideri aggiornare e il modello che hai usato per crearlo. Per esempio:

```
Description: Base Template to create Neptune Stack with Engine Version 1.2.0.1 using
 custom Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
```

```
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-cluster-parameter-group-description
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-parameter-group-description
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.1
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Aggiorna la proprietà `EngineVersion` da `1.2.0.1` a `1.2.0.2`:

```
Description: Template to upgrade minor engine version to 1.2.0.2
Parameters:
  DbInstanceType:
```

```
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-cluster-parameter-group-description
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-parameter-group-description
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Ora usalo per eseguire il modello AWS CloudFormation rivisto.

Esempio: aggiornamento della versione principale da 1.1.1.0 a 1.2.0.2 con gruppi di parametri predefiniti

Individua il `DBCluster` che desideri aggiornare e il modello che hai usato per crearlo. Per esempio:

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 default Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

- Aggiorna l'impostazione predefinita `DBClusterParameterGroup` a quella della famiglia di gruppi di parametri utilizzata dalla nuova versione del motore (qui `default.neptune1.2`).
- Per ogni `DBInstance` collegato al `DBCluster`, aggiorna l'impostazione predefinita `DBParameterGroup` a quella della famiglia utilizzata dalla nuova versione del motore (qui `default.neptune1.2`).
- Imposta la proprietà `DBInstanceParameterGroupName` sul gruppo di parametri predefinito nella famiglia (qui `default.neptune1.2`).

- Aggiorna la proprietà `EngineVersion` da `1.1.0.0` a `1.2.0.2`.

Il modello dovrebbe essere simile al seguente:

```
Description: Template to upgrade major engine version to 1.2.0.2 by using upgraded
 default parameter groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName: default.neptune1.2
      DBInstanceParameterGroupName: default.neptune1.2
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
```

Ora usa AWS CloudFormation per eseguire il modello rivisto.

Esempio: aggiornamento della versione principale da 1.1.1.0 a 1.2.0.2 con gruppi di parametri predefiniti

Individua il `DBCluster` che desideri aggiornare e il modello che hai usato per crearlo. Per esempio:

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 custom Parameter Groups
```

```
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Name: engineupgradetestcpg
      Family: neptune1
      Description: 'NeptuneDBClusterParameterGroup with family neptune1'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Name: engineupgradetestpg
      Family: neptune1
      Description: 'NeptuneDBParameterGroup1 with family neptune1'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
```

```
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

- Aggiorna la `DBClusterParameterGroup` famiglia personalizzata con quella utilizzata dalla nuova versione del motore `default.neptune1.2` (qui).
- Per ogni famiglia `DBInstance` allegata alla`DBCluster`, aggiorna la `DBParameterGroup` famiglia personalizzata a quella utilizzata dalla nuova versione del motore (qui`default.neptune1.2`).
- Imposta la proprietà `DBInstanceParameterGroupName` sul gruppo di parametri nella famiglia (qui `default.neptune1.2`).
- Aggiorna la proprietà `EngineVersion` da `1.1.0.0` a `1.2.0.2`.

Il modello dovrebbe essere simile al seguente:

```
Description: Template to upgrade major engine version to 1.2.0.2 by modifying existing
 custom parameter groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Name: engineupgradetestcpgnew
      Family: neptune1.2
      Description: 'NeptuneDBClusterParameterGroup with family neptune1.2'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Name: engineupgradetestpgnew
      Family: neptune1.2
      Description: 'NeptuneDBParameterGroup1 with family neptune1.2'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
```

```
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
      DBInstanceParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Ora AWS CloudFormation usatela per eseguire il modello rivisto.

Esempio: aggiornamento della versione principale da 1.1.1.0 a 1.2.0.2 con una combinazione di gruppi di parametri predefiniti e personalizzati

Individua il `DBCluster` che desideri aggiornare e il modello che hai usato per crearlo. Per esempio:

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 custom Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
```

```
    Properties:
      Family: neptune1
      Description: 'NeptuneDBClusterParameterGroup with family neptune1'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1
      Description: 'NeptuneDBParameterGroup with family neptune1'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  CustomNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
  DefaultNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
```

```
    Value:
      Ref: NeptuneDBCluster
```

- Per un gruppo di parametri del cluster personalizzato, aggiorna la famiglia `DBClusterParameterGroup` a quella corrispondente alla nuova versione del motore, vale a dire `neptune1.2`.
- Per un gruppo di parametri del cluster predefinito, aggiorna il `DBClusterParameterGroup` a quello corrispondente alla nuova versione del motore, vale a dire `default.neptune1.2`.
- Per ogni `DBInstance` collegato al `DBCluster`, aggiorna un gruppo predefinito `DBParameterGroup` a quello della famiglia utilizzato dalla nuova versione del motore (qui `default.neptune1.2`) e un gruppo di parametri personalizzato a uno che utilizza la famiglia supportata dalla nuova versione del motore (qui `neptune1.2`).
- Imposta la proprietà `DBInstanceParameterGroupName` sul gruppo di parametri della famiglia supportata dalla nuova versione del motore.

Il modello dovrebbe essere simile al seguente:

```
Description: Template to update Neptune Stack to Engine Version 1.2.0.1 using custom
 and default Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1.2
      Description: 'NeptuneDBClusterParameterGroup with family neptune1.2'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: 'NeptuneDBParameterGroup1 with family neptune1.2'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
```

```
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
      DBInstanceParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBClusterParameterGroup
  CustomNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
  DefaultNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Ora usa AWS CloudFormation per eseguire il modello rivisto.

Clonazione del database in Neptune

La clonazione dei database permette di creare cloni di tutti i database in modo rapido e conveniente in Amazon Neptune. I database clone richiedono soltanto uno spazio aggiuntivo minimo quando vengono creati per la prima volta. La clonazione di database utilizza un protocollo copy-on-write. I dati vengono copiati nel momento in cui vengono modificati, sui database di origine o sui database clone. Puoi creare più cloni dallo stesso cluster database. Puoi anche creare cloni ulteriori da altri cloni. Per ulteriori informazioni sul funzionamento del protocollo copy-on-write nel contesto dell'archiviazione Neptune, consulta Protocollo Copy-on-Write.

Puoi utilizzare la funzione di clonazione dei database in diversi casi d'uso, in particolare quando non vuoi compromettere l'ambiente di produzione, ad esempio:

- Sperimentare e valutare l'impatto delle modifiche, come cambiamenti degli schemi o dei gruppi di parametri
- Eseguire operazioni che utilizzano in modo intensivo i carichi di lavoro, come l'esportazione di dati o l'esecuzione di query analitiche.
- Creare una copia di un cluster database di produzione in un ambiente non di produzione per scopi di sviluppo o test.

Per creare un clone di un cluster database utilizzando la AWS Management Console

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, seleziona Instances (Istanze). Scegliere l'istanza principale per il cluster database del quale si desidera creare un clone.
3. Scegliere Instance actions (Operazioni istanza), quindi selezionare Create clone (Crea clone).
4. Nella pagina Create Clone (Crea clone), immettere un nome per l'istanza principale del cluster database clone come DB instance identifier (Identificatore istanze database).

   Se desiderato, configurare eventuali altre impostazioni per il cluster database clone. Per ulteriori informazioni sulle diverse impostazioni del cluster database, consultare Avvio mediante la console.
5. Scegliere Create Clone (Crea clone) per avviare il cluster database clone.

Per creare un clone di un cluster database utilizzando la AWS CLI

- Chiama il comando AWS CLI restore-db-cluster-to-point-in-time di Neptune e fornisci i seguenti valori:
  - `--source-db-cluster-identifier`: nome del cluster database di origine di cui creare un clone.
  - `--db-cluster-identifier`: nome del cluster database clone.
  - `--restore-type copy-on-write`: il valore `copy-on-write` indica che deve essere creato un cluster database clone.
  - `--use-latest-restorable-time`: questo valore indica che deve essere utilizzato il tempo di backup ripristinabile più recente.

> **Note**
>
> Il comando AWS CLI restore-db-cluster-to-point-in-time clona solo il cluster database e non le sue istanze database.

L'esempio Linux/UNIX seguente crea un clone dal cluster di database `source-db-cluster-id` e assegna il nome `db-clone-cluster-id` al clone.

```
aws neptune restore-db-cluster-to-point-in-time \
  --region us-east-1 \
  --source-db-cluster-identifier source-db-cluster-id \
  --db-cluster-identifier db-clone-cluster-id \
  --restore-type copy-on-write \
  --use-latest-restorable-time
```

Lo stesso esempio funziona su Windows se il carattere di escape di fine riga \ viene sostituito dall'equivalente ^ Windows:

```
aws neptune restore-db-cluster-to-point-in-time ^
  --region us-east-1 ^
  --source-db-cluster-identifier source-db-cluster-id ^
  --db-cluster-identifier db-clone-cluster-id ^
  --restore-type copy-on-write ^
  --use-latest-restorable-time
```

Limitazioni

La clonazione di database in Neptune presenta le seguenti limitazioni:

- Non puoi creare database clone tra regioni AWS. I database clone devono essere creati nella stessa regione dei database di origine.
- Un database clonato utilizza sempre la patch più recente della versione del motore Neptune utilizzata dal database da cui è stata clonata. Ciò vale anche se il database di origine non è stato ancora aggiornato a quella versione della patch. Tuttavia, la versione del motore non cambia.
- Al momento puoi effettuare un massimo di 15 cloni da una copia del tuo cluster database Neptune, inclusi quelli basati su altri cloni. Dopo aver raggiunto questo limite, devi creare un'altra copia del database, anziché clonarlo. Tuttavia, anche la nuova copia creata può avere fino a 15 cloni.
- La clonazione di database tra account diversi non è al momento supportata.
- Puoi fornire un diverso virtual private cloud (VPC) per il clone. Tuttavia, le sottoreti in quei VPC devono essere associate allo stesso gruppo di zone di disponibilità.

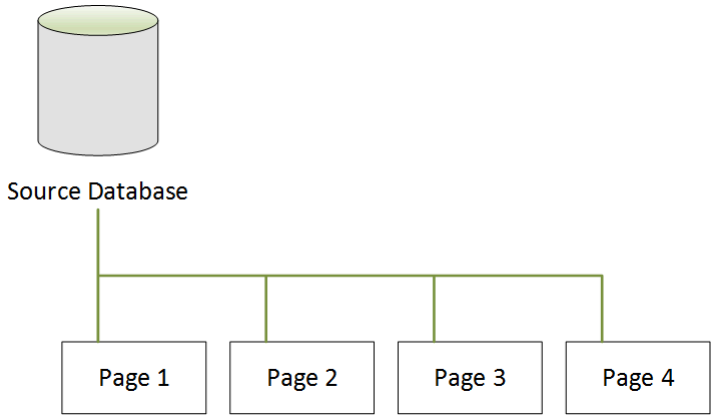
Protocollo Copy-on-Write per la clonazione di database

I seguenti scenari illustrano il funzionamento del protocollo di copia su scrittura.

- Database Neptune prima della clonazione
- Database Neptune dopo la clonazione
- Quando viene effettuata una modifica al database di origine
- Quando viene effettuata una modifica al database clone

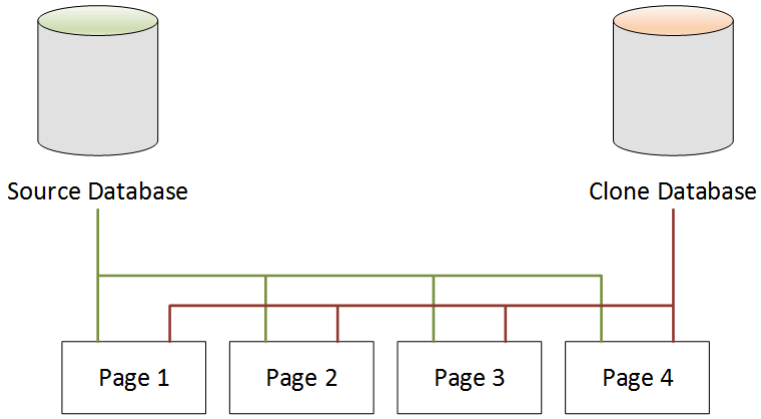
Database Neptune prima della clonazione

I dati in un database di origine vengono archiviati in pagine. Nel seguente grafico, il database di origine contiene quattro pagine.



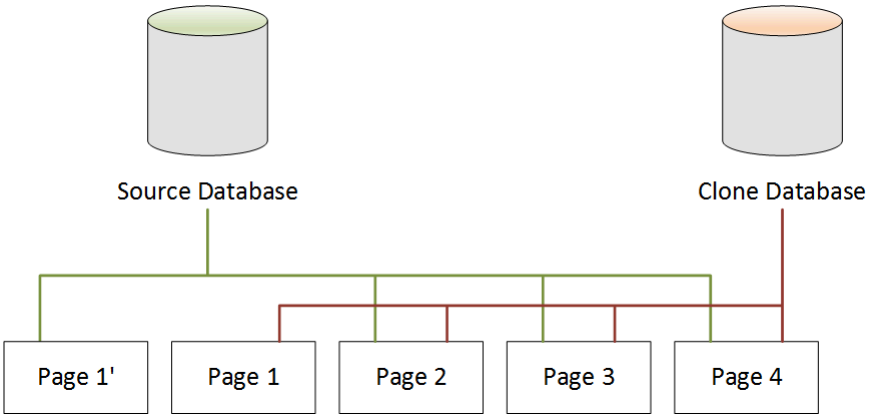
Database Neptune dopo la clonazione

Come mostrato nel grafico seguente, dopo la clonazione del database non vi sono modifiche nel database di origine. Entrambi i database di origine e clone rimandano alle stesse quattro pagine. Nessuna delle pagine è stata copiata fisicamente, quindi non è necessario ulteriore storage.



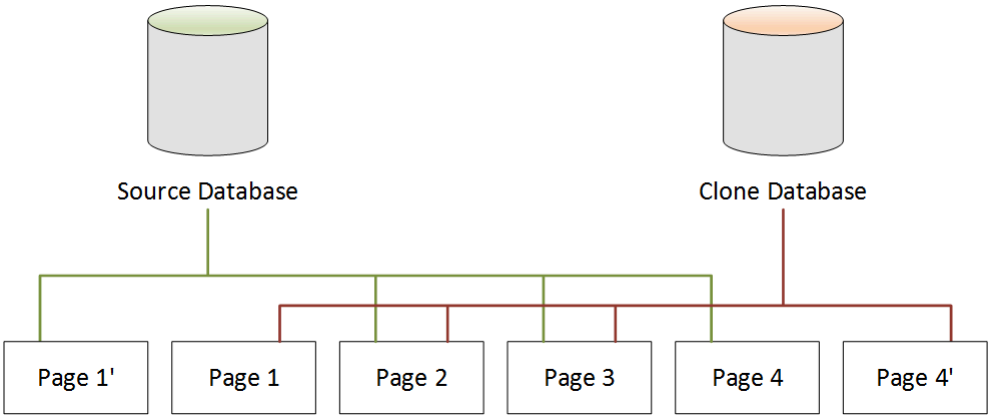
Quando viene effettuata una modifica al database di origine

Nell'esempio che segue il database di origine effettua una modifica ai dati a `Page 1`. Invece di scrivere nella `Page 1` originale, viene utilizzato storage aggiuntivo per creare una nuova pagina, denominata `Page 1'`. Il database di origine adesso rimanda alla nuova pagina `Page 1'` e anche a `Page 2`, `Page 3` e a `Page 4`. Il database clone continua a rimandare a `Page 1` fino a `Page 4`.



Quando viene effettuata una modifica al database clone

Nel grafico seguente anche il database clone ha effettuato una modifica, in questo caso a `Page 4`. Invece di scrivere nella `Page 4` originale, viene utilizzato storage aggiuntivo per creare una nuova pagina, denominata `Page 4'`. Il database di origine continua a rimandare a `Page 1'` e anche a `Page 2` fino alla `Page 4`, ma il database clone adesso rimanda a `Page 1` fino a `Page 3` e poi a `Page 4'`.



Come mostrato nel secondo scenario, dopo la clonazione del database non è richiesto storage aggiuntivo quando il clone viene creato. Tuttavia, in caso di modifiche nel database di origine e nel database clone, vengono create solo le pagine modificate, come abbiamo visto nel terzo e nel quarto scenario. Quando nel corso del tempo si verificano altre modifiche sia nel database di origine sia nel database clone, avrai bisogno in modo incrementale di più storage per acquisirle e archiviarle.

Eliminazione di un database di origine

L'eliminazione di un database di origine non influisce sui database clone associati a esso. I database clone continuano a rimandare alle pagine precedentemente di proprietà del database di origine.

Gestione delle istanze Amazon Neptune

Le seguenti sezioni contengono informazioni sulle operazioni a livello di istanza.

Argomenti

Classe di istanza espandibile T3 di Neptune.

Oltre alle classi di istanza a prestazioni fisse come R5 e R6, Amazon Neptune offre la possibilità di utilizzare un'istanza T3 a prestazioni espandibili. Mentre sviluppi un'applicazione a grafo, è importante che il database sia veloce e reattivo, anche se non lo devi utilizzare sempre. La classe di istanza db.t3.medium di Neptune è proprio ciò che dovresti usare in questa situazione, a un costo significativamente inferiore rispetto alla classe di istanza a prestazioni fisse meno costosa.

Un'istanza espandibile viene eseguita a un livello di base delle prestazioni della CPU fino a quando un carico di lavoro ha bisogno di più prestazioni e quindi si espande ben al di sopra della linea di base per tutto il tempo richiesto da un carico di lavoro. Il prezzo orario copre le espansioni, a condizione che l'utilizzo medio della CPU non superi la linea di base per un periodo di 24 ore. Per la maggior parte delle situazioni di sviluppo e test, ciò si traduce in buone prestazioni a basso costo.

Se inizi con una classe di istanza T3, puoi passare facilmente a una classe di istanza a prestazioni fisse quando è tutto pronto per passare alla fase di produzione utilizzando la AWS Management Console, AWS CLI o uno degli SDK AWS.

L'espansione T3 è governata da crediti CPU

Un credito CPU rappresenta l'utilizzo completo di una CPU virtuale core (vCPU) per un minuto. Ciò può anche tradursi in un utilizzo del 50% di una vCPU per due minuti o del 25% di due vCPU per due minuti e così via.

Un'istanza T3 accumula crediti CPU quando è inattiva e li usa quando è attiva, entrambi misurati alla risoluzione in millisecondi. La classe di istanza db.t3.medium ha due vCPU, ognuna delle quali guadagna 12 crediti CPU all'ora quando inattiva. Ciò significa che l'utilizzo del 20% di ogni vCPU determina un saldo di credito pari a zero della CPU. I 12 crediti CPU guadagnati sono spesi dal 20% di utilizzo della vCPU (dato che il 20% dei 60 minuti è 12). Questo utilizzo del 20% è quindi la percentuale di utilizzo di base che non produce né un saldo positivo né negativo di crediti CPU.

Il tempo di inattività (utilizzo della CPU inferiore al 20% del totale disponibile) fa sì che i crediti CPU vengano memorizzati in un bucket del saldo crediti, fino al limite per una classe di istanza db.t3.medium di 576 (il numero massimo di crediti CPU che potrebbero essere accumulati in 24 ore, ovvero 2 x 12 x 24). Oltre tale limite, i crediti CPU vengono semplicemente rimossi.

Se necessario, l'utilizzo della CPU può arrivare fino al 100% per tutto il tempo necessario da un carico di lavoro, anche quando il saldo dei crediti CPU scende al di sotto dello zero. Se l'istanza mantiene un saldo negativo continuamente per 24 ore, comporta un addebito aggiuntivo di $ 0,05 per

ogni -60 crediti CPU maturati in quel periodo. Per la maggior parte dei carichi di lavoro di sviluppo e test, l'espansione è solitamente coperta dal tempo di inattività prima o dopo l'espansione.

> **Note**
>
> La classe di istanza T3 di Neptune è configurata come la modalità illimitata di Amazon EC2.

Utilizzo dell'AWS Management Console per creare un'istanza espandibile T3

Nella AWS Management Console, puoi creare un'istanza del cluster database primario o un'istanza di replica di lettura che utilizza la classe di istanza `db.t3.medium` oppure puoi modificare un'istanza esistente per utilizzare la classe di istanza `db.t3.medium`.

Ad esempio, per creare una nuova istanza primaria del cluster database nella console Neptune:

- Scegliere Create Database (Crea database).
- Scegliere una versione del motore DB uguale o successiva a `1.0.2.2`.
- In Purpose (Scopo), scegliere Development and Testing (Sviluppo e test).
- Per la classe di istanza database, accetta il valore predefinito: `db.t3.medium — 2 vCPU, 4 GiB RAM`.

Utilizzo dell'AWS CLI per creare un'istanza espandibile T3

Puoi anche usare l'AWS CLI per effettuare la stessa operazione:

```
aws neptune create-db-cluster \
    --db-cluster-identifier (name for a new DB cluster) \
    --engine neptune \
    --engine-version "1.0.2.2"

aws neptune create-db-instance \
    --db-cluster-identifier (name of the new DB cluster) \
    --db-instance-identifier (name for the primary writer instance in the cluster) \
    --engine neptune \
    --db-instance-class db.t3.medium
```

Modifica di un'istanza database di Neptune (e applicazione immediata)

Puoi applicare immediatamente la maggior parte delle modifiche a un’istanza database di Amazon Neptune oppure posticiparle alla finestra di manutenzione successiva. Alcune modifiche, ad esempio le modifiche al gruppo di parametri, richiedono il riavvio manuale dell'istanza database per rendere effettiva la modifica.

> **Important**
>
> Le modifiche portano a un'interruzione se Neptune deve riavviare l'istanza database per applicarle. Esamina l'impatto sul database e sulle applicazioni prima di modificare le impostazioni dell'istanza database.

Impostazioni comuni e implicazioni relative ai tempi di inattività

La tabella seguente contiene informazioni su quali impostazioni puoi modificare, quando puoi applicare le modifiche e se le modifiche causano tempi di inattività per le istanze database.

Impostazioni dell'istanza database	Note sui tempi di inattività	
DB instance class (Classe istanza database)	Si verifica un'interruzione durante questa modifica, indipendentemente dal fatto che venga applicata immediatamente o nella finestra di manutenzione successiva.	
DB instance identifier (Identificatore istanze DB)	L'istanza database viene riavviata e si verifica un'interruzione durante questa modifica, indipendentemente dal fatto che venga applicata immediatamente o nella finestra di manutenzione successiva.	

Impostazioni dell'istanza database	Note sui tempi di inattività	
Subnet group (Gruppo di sottoreti)	L'istanza database viene riavviata e si verifica un'interruzione durante questa modifica, indipendentemente dal fatto che venga applicata immediatamente o nella finestra di manutenzione successiva.	
Gruppo di sicurezza	La modifica viene applicata in modo asincrono il prima possibile, indipendentemente da quando specifichi l'applicazione delle le modifiche, e non si verificano interruzioni.	–
Certificate Authority (Autorità di certificazione)	Per impostazione predefinita, l'istanza database viene riavviata quando si assegna una nuova autorità di certificazione.	
Database Port (Porta database)	La modifica avviene sempre immediatamente, il che causa il riavvio dell'istanza database e un'interruzione.	

Impostazioni dell'istanza database	Note sui tempi di inattività	
DB parameter group (Gruppo di parametri database)	La modifica di questa impostazione non comporta un'interruzione. Il nome del gruppo di parametri viene modificato immediatamente, ma le modifiche vere e proprie ai parametri vengono applicate solo al riavvio dell'istanza senza failover. In questo caso, l'istanza database non viene riavviata automaticamente e le modifiche ai parametri non vengono applicate nella finestra di manutenzione successiva. Tuttavia, se modifichi i parametri dinamici nel gruppo dei parametri del database appena associato , tali modifiche vengono applicate immediatamente senza eseguire il riavvio. Per ulteriori informazioni, consulta Riavvio di un'istanza database in Amazon Neptune.	
DB cluster parameter group (Gruppo di parametri del cluster database)	Il nome del gruppo di parametri di database viene modificato immediatamente.	

Impostazioni dell'istanza database	Note sui tempi di inattività	
Backup retention period (Periodo di retention dei backup)	Se specifichi che le modifiche devono essere apportate immediatamente, verranno applicate subito. In caso contrario, se cambi l'impostazione da un valore diverso da zero a un altro valore diverso da zero, la modifica viene applicata in modo asincrono, appena possibile. Qualsiasi altra modifica avviene durante la finestra di manutenzione successiva. Si verifica un'interruzione se cambi l'impostazione da zero a un valore diverso da zero o da un valore diverso da zero a zero.	
Log di audit	Seleziona Log di audit se vuoi utilizzare la registrazione dei log di audit mediante CloudWatch Logs. Devi anche impostare il parametro `neptune_enable_audit_log` nel gruppo di parametri del cluster database su `enable` (1) per abilitare la registrazione dei log di audit.	

Impostazioni dell'istanza database	Note sui tempi di inattività	
Auto minor version upgrade (Aggiornamento automatico della versione secondaria)	Seleziona Abilita l'aggiornamento automatico della versione secondaria se desideri che il cluster database Neptune riceva automaticamente aggiornamenti della versione secondaria del motore non appena diventano disponibili. L'opzione Aggiornamento automatico della versione secondaria si applica solo agli aggiornamenti delle versioni secondarie del motore per il cluster database Amazon Neptune. Non riguarda le patch normali applicate per mantenere la stabilità del sistema.	

Ridenominazione di un'istanza database di Neptune

Puoi rinominare un'istanza database di Amazon Neptune utilizzando la AWS Management Console. La ridenominazione di un'istanza database può avere ripercussioni significative. Di seguito è riportato un elenco di aspetti che è necessario valutare prima di ridenominare un'istanza database.

- Quando si rinomina un'istanza database, il rispettivo endpoint cambia, in quanto l'URL include il nome assegnato all'istanza. È sempre consigliabile reindirizzare il traffico dall'URL precedente a quello nuovo.
- Quando si rinomina un'istanza database, il nome DNS che veniva utilizzato precedentemente dall'istanza database viene eliminato immediatamente, ma può rimanere memorizzato nella cache per alcuni minuti. Il nuovo nome DNS per l'istanza database rinominata diventa effettivo dopo circa 10 minuti. L'istanza database ridenominata non è disponibile fino a quando il nuovo nome non diventa effettivo.
- Se stai ridenominando un'istanza, non puoi utilizzare un nome di istanza database esistente.
- Dopo che un'istanza database è stata ridenominata, tutte le repliche di lettura a essa associate rimangono tali. Ad esempio, supponiamo che disponi di un'istanza database utilizzata dal tuo database di produzione e che all'istanza siano associate diverse repliche di lettura. Se rinomini l'istanza database e quindi la sostituisci nell'ambiente di produzione con un snapshot di database, all'istanza database ridenominata restano comunque associate le repliche di lettura.
- Se riutilizzi un nome dell'istanza database, i parametri e gli eventi associati a tale nome vengono mantenuti. Ad esempio, se promuovi una replica di lettura e la rinomini assegnandole il nome dell'istanza primaria precedente, le metriche e gli eventi associati all'istanza primaria vengono associati all'istanza rinominata.
- I tag dell'istanza database rimangono con l'istanza, a prescindere dalla ridenominazione.
- Per un'istanza database ridenominata vengono mantenute le snapshot.

Per rinominare un'istanza database utilizzando la console Neptune

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, scegli Database.
3. Scegli il pulsante di opzione accanto all'istanza database da ridenominare.
4. Nel menu Instance actions (Operazioni istanza), scegli Modify (Modifica).

5. Inserisci un nuovo nome nella casella di testo DB instance identifier (Identificatore istanza database). Seleziona Apply immediately (Applica immediatamente), quindi scegli Continue (Continua).
6. Scegliere Modify DB instance (Modifica istanza database) per completare la modifica.

Riavvio di un'istanza database in Amazon Neptune

In alcuni casi, se modifichi un'istanza database Amazon Neptune, cambi il gruppo di parametri di database associato all'istanza o modifichi un parametro di database statico in un gruppo di parametri utilizzati dall'istanza, devi riavviare l'istanza per rendere effettive le modifiche.

Il riavvio di un'istanza database comporta il riavvio del servizio del motore di database. Con il riavvio, inoltre, vengono applicate all'istanza database tutte le modifiche al gruppo di parametri di database associato che erano in sospeso. Il riavvio di un'istanza database determina un guasto momentaneo dell'istanza, durante il quale lo stato dell'istanza database è impostato su riavvio in corso. Se l'istanza Neptune è configurata per Multi-AZ, il riavvio potrebbe essere eseguito attraverso un failover. Al completamento del riavvio, viene creato un evento Neptune.

Se l'istanza database è un'implementazione Multi-AZ, puoi imporre un failover da una zona di disponibilità a un'altra quando scegli l'opzione Reboot (Riavvia). Quando forzi un failover dell'istanza database, Neptune passa automaticamente a una replica di standby in un'altra zona di disponibilità. Quindi, aggiorna il record DNS per l'istanza database in modo che punti all'istanza database di standby. Di conseguenza, sarà necessario eliminare e ristabilire le connessioni esistenti alla propria istanza database.

L'opzione Reboot with failover (Riavvia con il failover) è utile quando desideri simulare un errore di un'istanza database per le operazioni di test e ripristino nella zona di disponibilità originale dopo un failover. Per ulteriori informazioni, consulta Configurazione e gestione di un'implementazione multi-AZ nella Guida per l'utente di Amazon RDS. Quando riavvii un cluster database, viene eseguito il failover nella replica di standby. Il riavvio di una replica Neptune non causa un failover.

Il tempo necessario per il riavvio è una funzione del processo di ripristino da arresto anomalo. Per ottimizzare i tempi del riavvio, consigliamo di ridurre il più possibile le attività del database durante il processo di riavvio per limitare l'attività di rollback per le transazioni in transito.

Nella console, l'opzione Reboot (Riavvia) potrebbe essere disabilitata se l'istanza database non si trova nello stato Available (Disponibile). Questo può dipendere da diversi motivi, ad esempio un backup in corso, una modifica richiesta dal cliente o un'operazione della finestra di manutenzione.

> **Note**
>
> Prima della Rilascio: 1.2.0.0 (21/07/2022), tutte le repliche di lettura in un cluster database venivano riavviate automaticamente al riavvio dell'istanza primaria (writer).

A partire dalla Rilascio: 1.2.0.0 (21/07/2022), il riavvio dell'istanza primaria non causa il riavvio di nessuna replica. Ciò significa che, se modifichi un parametro del cluster, devi riavviare ogni istanza separatamente per riprendere la modifica del parametro (vedi Gruppi di parametri).

Per riavviare un'istanza database utilizzando la console Neptune

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, scegli Database.
3. Scegliere l'istanza database che si desidera riavviare.
4. Scegli Instance actions (Operazioni istanza), quindi Reboot (Riavvia).
5. Per imporre un failover da una zona di disponibilità a un'altra, seleziona Reboot with failover? (Riavviare con failover?) nella finestra di dialogo Reboot DB Instance (Riavvia istanza database).
6. Scegliere Reboot (Riavvia). Per annullare il riavvio, scegliere Cancel (Annulla).

Eliminazione di un'istanza database in Amazon Neptune

Puoi eliminare un'istanza database di Amazon Neptune in qualsiasi stato e in qualsiasi momento, purché l'istanza sia stata avviata e la protezione da eliminazione sia disabilitata sull'istanza.

Impossibile eliminare un'istanza database se la protezione da eliminazione è abilitata

Puoi eliminare solo le istanze database per le quali è disabilitata la protezione da eliminazione. Neptune applica la protezione da eliminazione a prescindere dall'utilizzo della console, di AWS CLI o delle API per eliminare un'istanza database.

La protezione da eliminazione è abilitata per impostazione predefinita quando crei un'istanza database in produzione mediante la AWS Management Console.

La protezione da eliminazione è disabilitata per impostazione predefinita se utilizzi la AWS CLI o i comandi API per creare un'istanza database.

Per eliminare un'istanza database in cui è abilitata la protezione da eliminazione, modifica innanzitutto l'istanza per impostarne il campo `DeletionProtection` su `false`.

L'attivazione o la disattivazione della protezione da eliminazione non causa un'interruzione.

L'eliminazione di un'istanza database con snapshot finale

Per eliminare un'istanza database, devi specificare il nome dell'istanza e se desideri disporre di uno snapshot di database finale dell'istanza. Se l'istanza database che stai eliminando ha lo stato di Creating (Creazione), non potrai acquisire la snapshot DB finale. Se l'istanza database è in stato di errore con stato failed (non riuscito), incompatible-restore (incompatibile-ripristinare) o incompatible-network (incompatibile-rete), puoi eliminare l'istanza solo quando il parametro `SkipFinalSnapshot` è impostato su `true`.

Se elimini tutte le istanze database di Neptune in un cluster database utilizzando la AWS Management Console, il cluster database verrà eliminato automaticamente. Se utilizzi la AWS CLI oppure l'SDK, devi eliminare il cluster database manualmente dopo l'eliminazione dell'ultima istanza.

> **Important**
>
> Se elimini un intero cluster database, tutti i relativi backup automatici vengono eliminati contemporaneamente e non possono essere ripristinati. Di conseguenza, a meno che tu non scelga di creare manualmente uno snapshot di database finale, non potrai ripristinare lo stato

finale dell'istanza database in un secondo momento. Gli snapshot manuali di un'istanza non vengono eliminati quando viene eliminato il cluster.

Se l'istanza database che desideri eliminare è una replica di lettura, devi promuovere la replica di lettura o eliminarla.

In questo esempio, elimini un'istanza database con e senza snapshot di database finale.

L'eliminazione di un'istanza database senza snapshot finale

Se desideri eliminare rapidamente un'istanza database, puoi saltare la creazione di una snapshot DB finale. Quando elimini un'istanza database, tutti i backup automatici vengono eliminati e non possono essere recuperati. Gli snapshot manuali non vengono eliminati.

Per eliminare un'istanza database senza snapshot di database finale tramite la console Neptune

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, scegli Database.
3. Nell'elenco Instances (Istanze) scegli il pulsante di opzione accanto all'istanza database che desideri eliminare.
4. Scegli Instance actions (Operazioni istanza) e quindi Delete (Elimina).
5. Scegli No nel riquadro Create final snapshot? (Creare snapshot finale?).
6. Scegliere Delete (Elimina).

L'eliminazione di un'istanza database con snapshot finale

Se vuoi ripristinare un'istanza database eliminata in un secondo momento, puoi creare una snapshot DB finale. Tutti i backup automatici vengono eliminati e non possono essere recuperati. Gli snapshot manuali non vengono eliminati.

Per eliminare un'istanza database con snapshot di database finale tramite la console Neptune

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, scegli Database.

3. Nell'elenco Instances (Istanze) scegli il pulsante di opzione accanto all'istanza database che desideri eliminare.
4. Scegli Instance actions (Operazioni istanza) e quindi Delete (Elimina).
5. Scegli Yes (Sì) nel riquadro Create final snapshot? (Creare snapshot finale?).
6. Nel riquadro Final snapshot name (Nome snapshot finale) digita il nome della snapshot di database finale.
7. Scegliere Delete (Elimina).

Puoi controllare lo stato di un'istanza, stabilire il tipo di istanza, scoprire quale versione del rilascio del motore è installata attualmente e ottenere altre informazioni su un'istanza utilizzando l'API instance-status.

Amazon Neptune Serverless

Amazon Neptune Serverless è una soluzione di dimensionamento automatico on demand progettata per dimensionare i cluster database secondo necessità in modo da soddisfare anche aumenti considerevoli della domanda di elaborazione per poi ridurre nuovamente le risorse quando la domanda diminuisce. Consente di automatizzare i processi di monitoraggio del carico di lavoro e di regolazione della capacità del database Neptune. La capacità viene regolata automaticamente in base alla domanda dell'applicazione, pertanto viene addebitato solo l'importo per le risorse effettivamente necessarie per l'applicazione.

Casi d'uso per Neptune Serverless

Neptune Serverless supporta svariati tipi di carichi di lavoro. È adatto per carichi di lavoro impegnativi e altamente variabili e può essere molto utile se l'utilizzo del database è solitamente molto intenso per brevi periodi di tempo alternati a lunghi periodi di attività leggera o nessuna attività. Neptune Serverless è particolarmente utile per i seguenti casi d'uso:

- Carichi di lavoro variabili: carichi di lavoro che mostrano aumenti improvvisi e imprevedibili dell'attività della CPU. Con Neptune Serverless, il database a grafo dimensiona automaticamente la capacità in base alle esigenze del carico di lavoro e la riduce di nuovo quando il picco di attività è terminato. Così non dovrai più effettuare il provisioning per la capacità media o di picco. Puoi specificare un limite massimo di capacità per gestire i carichi di lavoro di picco; tale capacità viene utilizzata solo in caso di effettiva necessità.

  La granularità del dimensionamento garantita da Neptune Serverless consente di variare precisamente la capacità in base alle esigenze del carico di lavoro. Neptune Serverless può aggiungere o rimuovere capacità con incrementi granulari in base alle esigenze. Può aggiungere anche solo mezza unità di capacità Neptune (NCU, Neptune Capacity Unit) quando è richiesta una piccola quantità di capacità in più.
- Applicazioni multi-tenant: grazie a Neptune Serverless, puoi creare un cluster database separato per ciascuna delle applicazioni da eseguire, eliminando così la necessità di gestire i cluster tenant singolarmente. Ciascuno dei cluster tenant può avere periodi di attività e inattività diversi a seconda di svariati fattori, ma Neptune Serverless è in grado di dimensionarli in modo efficace senza il tuo intervento.
- Nuove applicazioni: quando implementi una nuova applicazione, spesso non sai con certezza quanta capacità servirà per il database. Utilizzando Neptune Serverless, puoi configurare un cluster

database in grado di eseguire il dimensionamento automatico per soddisfare i requisiti di capacità delle nuove applicazioni man mano che si sviluppano.

- Pianificazione della capacità: poniamo l'esempio che tu debba regolare la capacità del database o verificare la capacità ottimale del database in base al carico di lavoro. Per farlo, dovresti modificare le classi di tutte le istanze database in un cluster. Neptune Serverless ti permette di evitare questo sovraccarico amministrativo. Invece puoi modificare le istanze database con provisioning esistenti in istanze serverless oppure le istanze serverless in istanze con provisioning senza dover creare una nuova istanza o un nuovo cluster database.
- Sviluppo e test: Neptune Serverless è perfetto anche per ambienti di sviluppo e test. Con Neptune Serverless puoi creare istanze di database con una capacità massima sufficientemente alta da permetterti di testare la tua applicazione più esigente e una capacità minima bassa per tutte le altre situazioni in cui il sistema potrebbe restare inattivo tra un test e l'altro.

Neptune Serverless dimensiona solo la capacità di calcolo. Il volume di archiviazione rimane lo stesso e non è influenzato dal dimensionamento serverless.

> **Note**
>
> Puoi anche utilizzare il dimensionamento automatico di Neptune con Neptune Serverless per gestire diversi tipi di variazioni del carico di lavoro.

Vincoli di Amazon Neptune Serverless

- Non disponibile in tutte le regioni. Neptune Serverless è disponibile solo nelle seguenti regioni:
  - Stati Uniti orientali (Virginia settentrionale): `us-east-1`
  - Stati Uniti orientali (Ohio): `us-east-2`
  - Stati Uniti occidentali (California settentrionale): `us-west-1`
  - Stati Uniti occidentali (Oregon): `us-west-2`
  - Canada (Centrale): `ca-central-1`
  - Europa (Stoccolma): `eu-north-1`
  - Europa (Irlanda): `eu-west-1`
  - Europa (Londra): `eu-west-2`
  - Europa (Francoforte): `eu-central-1`

  - Asia Pacifico (Tokyo): `ap-northeast-1`
  - Asia Pacifico (Singapore): `ap-southeast-1`
  - Asia Pacifico (Sydney): `ap-southeast-2`
- Non disponibile nelle versioni meno recenti del motore. Neptune Serverless è disponibile solo nei rilasci del motore 1.2.0.1 o successivi.
- Non compatibile con la cache di ricerca di Neptune. La cache di ricerca non è compatibile con le istanze database serverless.
- La memoria massima in un'istanza serverless è di 256 GB. L'impostazione di `MaxCapacity` su 128 NCU (l'impostazione massima supportata) consente a un'istanza Neptune Serverless di dimensionare fino a 256 GB di memoria, che equivale a quella di un tipo di istanza `R6g.8XL` con provisioning.

Dimensionamento della capacità in un cluster database Neptune Serverless

La configurazione di un cluster database Neptune Serverless è simile alla configurazione di un normale cluster con provisioning, con una configurazione aggiuntiva per le unità minime e massime per il dimensionamento, e con il tipo di istanza impostato su `db.serverless`. La configurazione del dimensionamento è definita in unità di capacità Neptune (NCU, Neptune Capacity Unit), ognuna delle quali include 2 GiB (gibibyte) di memoria (RAM) insieme alle reti e alla capacità del processore virtuale (vCPU) associate. L'impostazione avviene come parte di un oggetto `ServerlessV2ScalingConfiguration`, rappresentato in JSON in questo modo:

```
"ServerlessV2ScalingConfiguration": {
  "MinCapacity": (minimum NCUs, a floating-point number such as   1.0),
  "MaxCapacity": (maximum NCUs, a floating-point number such as 128.0)
}
```

In qualsiasi momento, ogni istanza di scrittura o lettura di Neptune ha una capacità misurata da un numero in virgola mobile che rappresenta il numero di NCU attualmente utilizzate dall'istanza in questione. Puoi utilizzare la CloudWatch ServerlessDatabaseCapacitymetrica a livello di istanza per scoprire quante NCU sta attualmente utilizzando una determinata istanza DB e la metrica NCUUtilization per scoprire quale percentuale della sua capacità massima sta utilizzando l'istanza. Entrambe le metriche sono disponibili anche a livello di cluster database per mostrare l'utilizzo medio delle risorse per il cluster database nel suo complesso.

Quando crei un cluster database Neptune Serverless, imposti il numero minimo e massimo di unità di capacità Neptune (NCU) per tutte le istanze serverless.

Il valore NCU minimo specificato determina le dimensioni minime fino alle quali può ridursi un'istanza serverless del cluster database e, allo stesso modo, il valore NCU massimo stabilisce le dimensioni massime fino alle quali può accrescersi un'istanza serverless. Il valore NCU massimo che è possibile impostare è 128,0 NCU e il valore minimo è 1,0 NCU.

Neptune tiene costantemente traccia del carico su ciascuna istanza Neptune Serverless monitorandone l'utilizzo di risorse come CPU, memoria e rete. Il carico viene generato dalle operazioni del database dell'applicazione, dall'elaborazione in background per il server e da altre attività amministrative.

Quando il carico su un'istanza serverless raggiunge il limite della capacità attuale oppure quando Neptune rileva altri problemi di prestazioni, l'istanza aumenta automaticamente le proprie dimensioni. Quando il carico sull'istanza diminuisce, la capacità si riduce fino al valore NCU minimo configurato, con la capacità della CPU che viene rilasciata prima della memoria. Questa architettura consente il rilascio controllato e graduale delle risorse, oltre a gestire in modo efficace le fluttuazioni della domanda.

È possibile dimensionare un'istanza di lettura insieme all'istanza di scrittura oppure eseguire il dimensionamento separatamente impostando il livello di promozione per le istanze. Le istanze di lettura con livello di promozione 0 e 1 vengono dimensionate congiuntamente a quella di scrittura; in questo modo, la capacità delle istanze rimane delle dimensioni corrette per assumere rapidamente il controllo del carico di lavoro dall'istanza di scrittura in caso di failover. Le istanze di lettura con livello di promozione da 2 a 15 vengono dimensionate separatamente l'una dall'altra e anche dall'istanza di scrittura.

Se hai creato un cluster database Neptune come cluster Multi-AZ per garantire un'elevata disponibilità, Neptune Serverless aumenta e riduce le dimensioni delle istanze in tutte le zone di disponibilità (AZ) in base al carico del database. Puoi impostare il livello di promozione di un'istanza di lettura in una zona di disponibilità secondaria su 0 o 1 affinché le dimensioni dell'istanza aumentino o diminuiscano in base alla capacità dell'istanza di scrittura nella zona di disponibilità principale, così che l'istanza di lettura sia pronta ad assumere il controllo del carico di lavoro attuale in qualsiasi momento.

> **Note**
>
> L'archiviazione per un cluster database Neptune prevede sei copie di tutti i dati, distribuite su tre zone di disponibilità, indipendentemente dal fatto che il cluster sia stato creato o meno come cluster Multi-AZ. La replica dell'archiviazione è gestita dal sottosistema di archiviazione e non è influenzata da Neptune Serverless.

Scelta di un valore della capacità minima per un cluster database Neptune Serverless

Il valore minimo che puoi impostare per la capacità minima è `1.0` NCU.

Assicurati di non impostare un valore minimo inferiore a quello richiesto dall'applicazione per funzionare in modo efficiente. Impostare un valore troppo basso può comportare una maggiore frequenza di timeout in determinati carichi di lavoro ad alto consumo di memoria.

Impostare il valore minimo più basso possibile può far risparmiare denaro, poiché in tal modo il cluster usa la quantità minima di risorse quando la domanda è bassa. Tuttavia, se il carico di lavoro tende a fluttuare drasticamente (da molto basso a molto alto), potresti voler impostare un valore minimo più alto, poiché così facendo le dimensioni delle istanze Neptune Serverless aumenterebbero più velocemente.

Il motivo è che Neptune sceglie gli incrementi di dimensionamento in base alla capacità attuale. Se la capacità attuale è bassa, all'inizio Neptune aumenta le dimensioni lentamente. Con un valore minimo più alto, Neptune inizia con un incremento di dimensionamento maggiore e può quindi aumentare le dimensioni più velocemente per far fronte a un aumento notevole e improvviso del carico di lavoro.

Scelta di un valore della capacità massima per un cluster database Neptune Serverless

Il valore massimo che puoi impostare per la capacità massima è `128.0` NCU, mentre il valore minimo che puoi impostare per la capacità massima è `2.5` NCU. Il valore della capacità massima (qualsiasi esso sia) deve essere pari almeno al valore impostato per la capacità minima.

Come regola generale, imposta un valore della capacità massima abbastanza alto da poter gestire il carico di picco che l'applicazione potrebbe riscontrare. Impostare un valore troppo basso può

comportare una maggiore frequenza di timeout in determinati carichi di lavoro ad alto consumo di memoria.

Il vantaggio di impostare il valore più alto possibile per la capacità massima è che l'applicazione sarà probabilmente in grado di gestire anche i carichi di lavoro più imprevisti. Lo svantaggio invece consiste nel fatto che si perde in una certa misura la capacità di prevedere e controllare i costi delle risorse. Un picco imprevisto nella domanda può determinare costi nettamente superiori a quelli previsti dal budget.

Il vantaggio di un valore della capacità massima attentamente mirato è che consente di soddisfare i picchi di domanda, ponendo al contempo un limite ai costi di calcolo di Neptune.

> **Note**
>
> La modifica dell'intervallo di capacità di un cluster database Neptune Serverless comporta modifiche ai valori predefiniti di alcuni parametri di configurazione. Neptune può applicare immediatamente alcune di queste nuove impostazioni predefinite, ma alcune modifiche ai parametri dinamici diventano effettive solo dopo un riavvio. Lo stato `pending-reboot` indica che è necessario un riavvio per applicare le modifiche ad alcuni parametri.

Uso della configurazione esistente per stimare i requisiti serverless

Se in genere modifichi la classe delle istanze di database con provisioning per soddisfare carichi di lavoro eccezionalmente elevati o ridotti, puoi utilizzare tale esperienza per calcolare una stima approssimativa dell'equivalente intervallo di capacità per Neptune Serverless.

Calcolo di una stima per l'impostazione della capacità minima più appropriata

Puoi applicare le tue conoscenze relative al cluster database Neptune esistente per stimare l'impostazione della capacità minima serverless che funzionerà meglio.

Ad esempio, se un carico di lavoro con provisioning ha requisiti di memoria troppo elevati per classi di istanze database di piccole dimensioni come `T3` o `T4g`, scegli un'impostazione NCU minima che fornisca memoria paragonabile a una classe di istanza database `R5` o `R6g`.

Altrimenti, supponiamo di utilizzare la classe di istanza database `db.r6g.xlarge` quando un cluster ha un carico di lavoro ridotto. Questa classe di istanza database ha 32 GiB di memoria, quindi

è possibile specificare un'impostazione NCU minima pari a 16 per creare istanze serverless che possono ridurre le proprie dimensioni approssimativamente fino alla stessa capacità (ogni NCU corrisponde a circa 2 GiB di memoria). Se talvolta l'istanza `db.r6g.xlarge` è sottoutilizzata, potresti essere in grado di specificare un valore inferiore.

Se l'applicazione funziona nel modo più efficiente quando le istanze di database possono contenere una determinata quantità di dati nella memoria o nella cache del buffer, valuta la possibilità di specificare un'impostazione NCU minima sufficientemente alta da garantire memoria sufficiente. In caso contrario, i dati potrebbero essere espulsi dalla cache del buffer quando le istanze serverless riducono le proprie dimensioni e devono essere riletti nella cache del buffer nel corso del tempo quando le istanze tornano ad aumentare le proprie dimensioni. Se la quantità di I/O per riportare i dati nella cache del buffer è rilevante, potrebbe essere più efficace scegliere un valore NCU minimo più alto.

Se ritieni che le istanze serverless funzionino per la maggior parte del tempo a una determinata capacità, è consigliabile impostare la capacità minima su un valore leggermente inferiore a quello. Neptune Serverless può calcolare in modo più efficace una stima della quantità e della velocità dell'aumento di dimensioni quando la capacità attuale non è sensibilmente inferiore alla capacità richiesta.

In una configurazione mista, con un'istanza di scrittura con provisioning e più istanze di lettura Neptune Serverless, queste ultime non vengono dimensionate congiuntamente all'istanza di scrittura. Poiché le istanze si dimensionano in modo indipendente, l'impostazione di una capacità minima ridotta può comportare un eccessivo ritardo di replica. La capacità potrebbe non essere sufficiente per tenere il passo con le modifiche apportate dall'istanza di scrittura quando il carico di lavoro è ad alta intensità di scrittura. In questa situazione, imposta una capacità minima paragonabile a quella dell'istanza di scrittura. In particolare, se osservi un ritardo di replica nelle istanze di lettura che si trovano nei livelli di promozione da 2 a 15, è opportuno che tu aumenti l'impostazione della capacità minima per il cluster.

Calcolo di una stima per l'impostazione della capacità massima più appropriata

Puoi anche applicare le tue conoscenze relative al cluster database Neptune esistente per stimare l'impostazione della capacità massima serverless che funzionerà meglio.

Ad esempio, supponiamo di utilizzare la classe di istanza database `db.r6g.4xlarge` quando un cluster ha un carico di lavoro elevato. Questa classe di istanza database ha 128 GiB di memoria, quindi è possibile specificare un'impostazione NCU massima pari a 64 per configurare istanze Neptune Serverless equivalenti (ogni NCU corrisponde a circa 2 GiB di memoria). Puoi specificare

un valore più elevato per consentire all'istanza database di aumentare ulteriormente le proprie dimensioni nel caso in cui l'istanza `db.r6g.4xlarge` non riesca sempre a gestire il carico di lavoro.

Se i picchi imprevisti nel carico di lavoro sono rari, può essere opportuno impostare la capacità massima su un valore abbastanza elevato da mantenere invariate le prestazioni dell'applicazione anche durante tali picchi. D'altra parte, potresti voler impostare una capacità massima inferiore che consenta di ridurre la velocità di trasmissione effettiva durante i picchi insoliti, ma che al contempo permetta a Neptune di gestire i carichi di lavoro previsti senza problemi e con costi limitati.

Configurazione aggiuntiva per istanze e cluster database Neptune Serverless

Oltre a impostare la capacità minima e massima per il cluster database Neptune Serverless, ci sono alcune altre opzioni di configurazione da considerare.

Combinazione di istanze serverless e istanze con provisioning in un cluster database

Un cluster database non deve necessariamente essere solo serverless: puoi creare una combinazione di istanze serverless e istanze con provisioning (ossia una configurazione mista).

Ad esempio, supponi di aver bisogno di più capacità in scrittura rispetto a quella disponibile in un'istanza serverless. In questo caso, puoi configurare il cluster con un'istanza di scrittura con provisioning di dimensioni molto grandi e continuare a utilizzare le istanze serverless per le istanze di lettura.

Altrimenti supponi che il carico di lavoro in scrittura per il cluster vari, ma che il carico di lavoro in lettura sia costante. In questo caso, puoi configurare il cluster con un'istanza di scrittura serverless e una o più istanze di lettura con provisioning.

Per informazioni su come creare un cluster database con configurazione mista, consulta la pagina Utilizzo di Amazon Neptune Serverless.

Impostazione dei livelli di promozione per le istanze Neptune Serverless

Per cluster contenenti più istanze serverless o una combinazione di istanze serverless e istanze con provisioning, presta attenzione all'impostazione del livello di promozione per ciascuna istanza

serverless. Questa impostazione ha un controllo maggiore sul comportamento delle istanze serverless rispetto a quello che ha sulle istanze database con provisioning.

In AWS Management Console, si specifica questa impostazione utilizzando la priorità di failover in Configurazione aggiuntiva nelle pagine Crea database, Modifica istanza e Aggiungi lettore. Questa proprietà viene visualizzata per le istanze esistenti nella colonna facoltativa Livello di priorità nella pagina Database. È inoltre possibile visualizzare questa proprietà nella pagina dei dettagli di un cluster database o un'istanza.

Per le istanze con provisioning, la scelta del livello 0-15 determina solo l'ordine in base al quale Neptune sceglie l'istanza di lettura da promuovere a istanza di scrittura durante un'operazione di failover. Per le istanze di lettura Neptune Serverless, il numero del livello determina anche se l'istanza aumenta le proprie dimensioni in modo che corrispondano alla capacità dell'istanza di scrittura oppure se si dimensiona separatamente da essa in base solo al proprio carico di lavoro.

Le istanze di lettura Neptune Serverless di livello 0 o 1 vengono mantenute a una capacità minima pari almeno a quella dell'istanza di scrittura, in modo che siano pronte a sostituire quest'ultima in caso di failover. Se l'istanza di scrittura è con provisioning, Neptune stima la capacità serverless equivalente e utilizza tale stima come capacità minima per l'istanza di lettura serverless.

Le istanze di lettura Neptune Serverless di livello da 2 a 15 non hanno lo stesso vincolo per quanto riguarda la capacità minima ed eseguono il dimensionamento indipendentemente dall'istanza di scrittura. Quando sono inattive, riducono le proprie dimensioni fino al valore NCU minimo specificato nell'intervallo di capacità del cluster. Tuttavia, ciò può causare problemi se il carico di lavoro in lettura aumenta molto e rapidamente.

Mantenere la capacità in lettura allineata a quella in scrittura

Una cosa importante da tenere a mente è che l'obiettivo è quello di garantire che le istanze di lettura possano tenere il passo con l'istanza di scrittura, onde evitare un ritardo di replica eccessivo. Questo è un aspetto a cui prestare particolare attenzione in due situazioni, in cui le istanze di lettura serverless non si dimensionano automaticamente in modo sincronizzato con l'istanza di scrittura:

- Quando l'istanza di scrittura è con provisioning e le istanze di lettura sono serverless.
- Quando l'istanza di scrittura è serverless e le istanze di lettura serverless hanno un livello di promozione compreso tra 2 e 15.

In entrambi i casi, è opportuno impostare la capacità serverless minima in modo che corrisponda a quella di scrittura prevista, così da garantire che le operazioni di lettura non vadano in timeout

causando potenziali riavvii. Nel caso di un'istanza di scrittura con provisioning, è opportuno impostare la capacità minima in modo che corrisponda a quella dell'istanza con provisioning. Nel caso di un'istanza di scrittura serverless, l'impostazione ottimale potrebbe essere più difficile da prevedere.

Poiché l'intervallo di capacità delle istanze è impostato a livello di cluster, tutte le istanze serverless sono controllate dalle stesse impostazioni di capacità minima e massima. Le istanze di lettura di livello 0 e 1 si dimensionano in modo sincronizzato con l'istanza di scrittura, mentre le istanze con livello di promozione compreso tra 2 e 15 si dimensionano indipendentemente l'una dall'altra e dall'istanza di scrittura, in funzione del rispettivo carico di lavoro. Se imposti la capacità minima su un valore troppo basso, le istanze inattive di livello da 2 a 15 possono ridursi a dimensioni troppo piccole, che non riescono ad aumentare di nuovo abbastanza velocemente da gestire un'improvviso picco nell'attività di scrittura.

Evitare di impostare un valore di timeout troppo elevato

Potrebbero essere applicati costi imprevisti se viene impostato un valore di timeout delle query troppo elevato su un'istanza serverless.

Senza un'impostazione di timeout ragionevole, è possibile eseguire inavvertitamente una query che richiede un tipo di istanza potente e costoso, e che continua ad essere eseguita per un periodo di tempo molto lungo, con costi mai previsti. Per evitare questa situazione, specifica un valore di timeout delle query adatto alla maggior parte delle query e che causa il timeout solo delle query con tempi di esecuzione inaspettatamente lunghi.

Questo vale sia per i valori di timeout generali delle query, impostati utilizzando i parametri, sia per i valori di timeout per specifiche query, impostati utilizzando gli hint di query.

Ottimizzazione della configurazione di Neptune Serverless

Se il cluster database Neptune Serverless non è ottimizzato per il carico di lavoro in esecuzione, potresti notare un funzionamento non ottimale. Puoi regolare l'impostazione della capacità minima e/o massima in modo che il dimensionamento avvenga senza problemi di memoria.

- Incremento dell'impostazione minima della capacità per il cluster Questa operazione consente di correggere la situazione in cui un'istanza inattiva torna a una capacità con meno memoria di quella richiesta dall'applicazione e dalle funzionalità abilitate.
- Incremento dell'impostazione massima della capacità per il cluster Questa operazione consente di correggere la situazione in cui un database occupato non è in grado di aumentare le dimensioni

fino a una capacità con memoria sufficiente per gestire il carico di lavoro ed eventuali funzionalità abilitate che consumano molta memoria.

- Modifica il carico di lavoro sull'istanza in questione. Ad esempio, puoi aggiungere istanze di lettura al cluster per distribuire il carico in lettura su più istanze.
- Ottimizza le query dell'applicazione in modo che utilizzino meno risorse.
- Prova a utilizzare un'istanza con provisioning che abbia dimensioni più grandi del numero massimo di NCU disponibili in Neptune Serverless, per vedere se è più adatta ai requisiti di memoria e CPU del carico di lavoro.

Utilizzo di Amazon Neptune Serverless

Puoi creare un nuovo cluster database Neptune come cluster serverless oppure, in alcuni casi, puoi convertire un cluster database esistente per utilizzarlo come serverless. Puoi anche convertire le istanze database di un cluster database serverless da e in istanze serverless. Puoi usare Neptune Serverless solo in uno dei Regioni AWS paesi in cui è supportato, con alcune altre limitazioni (vedi). Vincoli di Amazon Neptune Serverless

Per creare un cluster database Neptune Serverless, puoi anche usare lo stack AWS CloudFormation Neptune.

Creazione di un nuovo cluster database che utilizza Serverless

Per creare un cluster database Neptune che utilizza Serverless, è possibile utilizzare la AWS Management Console come per creare un cluster con provisioning. La differenza consiste nel fatto che, in base alle dimensioni dell'istanza database, devi impostare la classe di istanza database su serverless. Quando esegui questa procedura, devi impostare l'intervallo di capacità serverless per il cluster.

Puoi anche creare un cluster DB serverless usando i comandi AWS CLI with come questi (su Windows, sostituisci '\' con '^'):

```
aws neptune create-db-cluster \
  --region (an Regione AWS region that supports serverless) \
  --db-cluster-identifier (ID for the new serverless DB cluster) \
  --engine neptune \
  --engine-version (optional: 1.2.0.1 or above) \
  --serverless-v2-scaling-configuration "MinCapacity=1.0, MaxCapacity=128.0"
```

Puoi anche specificare il parametro `serverless-v2-scaling-configuration` in questo modo:

```
  --serverless-v2-scaling-configuration '{"MinCapacity":1.0, "MaxCapacity":128.0}'
```

Quindi, puoi eseguire il comando `describe-db-clusters` per l'attributo `ServerlessV2ScalingConfiguration`, che deve restituire le impostazioni dell'intervallo di capacità specificate:

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": (the specified minimum number of NCUs),
    "MaxCapacity": (the specified maximum number of NCUs)
}
```

Conversione di un'istanza o un cluster database esistente in Serverless

Se hai un cluster database Neptune che utilizza la versione del motore 1.2.0.1 o successiva, puoi convertirlo in serverless. Questo processo comporta un certo tempo di inattività.

Il primo passaggio prevede di aggiungere un intervallo di capacità al cluster esistente. Puoi farlo usando o usando un AWS CLI comando come questo (in Windows, sostituisci '\' con '^'): AWS Management Console

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (your DB cluster ID) \
  --serverless-v2-scaling-configuration \
      MinCapacity=(minimum number of NCUs, such as  2.0), \
      MaxCapacity=(maximum number of NCUs, such as 24.0)
```

Il passaggio successivo prevede di creare una nuova istanza di database serverless per sostituire l'istanza principale esistente (quella di scrittura) nel cluster. Ancora una volta, puoi eseguire questa operazione e tutti i passaggi successivi utilizzando il AWS Management Console o il AWS CLI. In entrambi i casi, devi specificare la classe di istanza database come serverless. Il AWS CLI comando sarebbe simile al seguente (su Windows, sostituisci '\' con '^'):

```
aws neptune create-db-instance \
  --db-instance-identifier (an instance ID for the new writer instance) \
  --db-cluster-identifier (ID of the DB cluster) \
  --db-instance-class db.serverless
  --engine neptune
```

Una volta che la nuova istanza di scrittura è diventata disponibile, esegui un failover per impostarla come istanza di scrittura per il cluster:

```
aws neptune failover-db-cluster \
  --db-cluster-identifier (ID of the DB cluster) \
  --target-db-instance-identifier (instance ID of the new serverless instance)
```

Quindi, elimina la vecchia istanza di scrittura:

```
aws neptune delete-db-instance \
  --db-instance-identifier (instance ID of the old writer instance) \
  --skip-final-snapshot
```

Infine, segui la stessa procedura per creare una nuova istanza serverless che sostituisca ogni istanza di lettura con provisioning esistente che desideri trasformare in un'istanza serverless ed elimina le istanze con provisioning esistenti (non è necessario eseguire un failover per le istanze di lettura).

Modifica dell'intervallo di capacità di un cluster database serverless esistente

Puoi modificare l'intervallo di capacità di un cluster database Neptune Serverless utilizzando l'interfaccia AWS CLI in questo modo (in Windows, sostituisci "\" con "^"):

```
aws neptune modify-db-cluster \
  --region (an AWS region that supports serverless) \
  --db-cluster-identifier (ID of the serverless DB cluster) \
  --apply-immediately \
  --serverless-v2-scaling-configuration MinCapacity=4.0, MaxCapacity=32
```

La modifica dell'intervallo di capacità comporta modifiche ai valori di default di alcuni parametri di configurazione. Neptune può applicare immediatamente alcune di queste nuove impostazioni predefinite, ma alcune modifiche ai parametri dinamici diventano effettive solo dopo un riavvio. Lo stato `pending-reboot` indica che è necessario un riavvio per applicare le modifiche ad alcuni parametri.

Modifica di un'istanza di database Serverless in istanza con provisioning

Tutto ciò che devi fare per convertire un'istanza Neptune Serverless in un'istanza con provisioning è cambiare la relativa classe di istanza impostandola su una delle classi di istanza con provisioning. Per informazioni, consulta Modifica di un'istanza database di Neptune (e applicazione immediata).

Monitoraggio della capacità serverless con Amazon CloudWatch

È possibile CloudWatch utilizzarlo per monitorare la capacità e l'utilizzo delle istanze serverless Neptune nel cluster DB. Esistono due CloudWatch metriche che consentono di tenere traccia dell'attuale capacità serverless sia a livello di cluster che a livello di istanza:

- **`ServerlessDatabaseCapacity`**: come metrica a livello di istanza, `ServerlessDatabaseCapacity` restituisce la capacità attuale dell'istanza, espressa in NCU. Come metrica a livello di cluster, restituisce la media di tutti i valori `ServerlessDatabaseCapacity` di tutte le istanze database nel cluster.
- **`NCUUtilization`**: questa metrica restituisce la percentuale della capacità possibilmente in uso. Viene calcolata come valore `ServerlessDatabaseCapacity` attuale (a livello di istanza o a livello di cluster) diviso per l'impostazione della capacità massima per il cluster database.

  Se questa metrica si avvicina al 100% a livello di cluster (il che significa che il cluster ha le dimensioni massime possibili), valuta la possibilità di incrementare l'impostazione della capacità massima.

  Se si avvicina al 100% per un'istanza di lettura mentre l'istanza di scrittura non è vicina alla capacità massima, prendi in considerazione l'aggiunta di altre istanze di lettura per distribuire il carico di lavoro in lettura.

Tieni presente che le metriche `CPUUtilization` e `FreeableMemory` hanno significati leggermente diversi per le istanze serverless rispetto a quello delle istanze con provisioning. In un contesto serverless, `CPUUtilization` è una percentuale calcolata come la quantità di CPU attualmente in uso divisa per la quantità di CPU disponibile alla capacità massima. In modo analogo, `FreeableMemory` restituisce la quantità di memoria che è possibile liberare e che sarebbe disponibile se un'istanza avesse la capacità massima.

L'esempio seguente mostra come utilizzare AWS CLI su Linux per recuperare i valori di capacità minima, massima e media per una determinata istanza DB, misurati ogni 10 minuti nell'arco di un'ora. Il comando Linux `date` specifica l'ora di inizio e l'ora di fine rispetto alla data e all'ora correnti. La

funzione `sort_by` nel parametro `--query` dispone i risultati in ordine cronologico, in base al campo `Timestamp`:

```
aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '1 hour ago')" \
  --end-time "$(date -d 'now')" \
  --period 600 \
  --namespace "AWS/Neptune"
  --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=(instance ID) \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
  --output table
```

Acquisizione di modifiche al grafo in tempo reale mediante Neptune Streams

Neptune Streams registra automaticamente ogni modifica al grafo, nell'ordine in cui viene eseguita, in modo completamente gestito. Dopo aver abilitato la funzionalità Streams, Neptune si occupa di disponibilità, backup, sicurezza e scadenza.

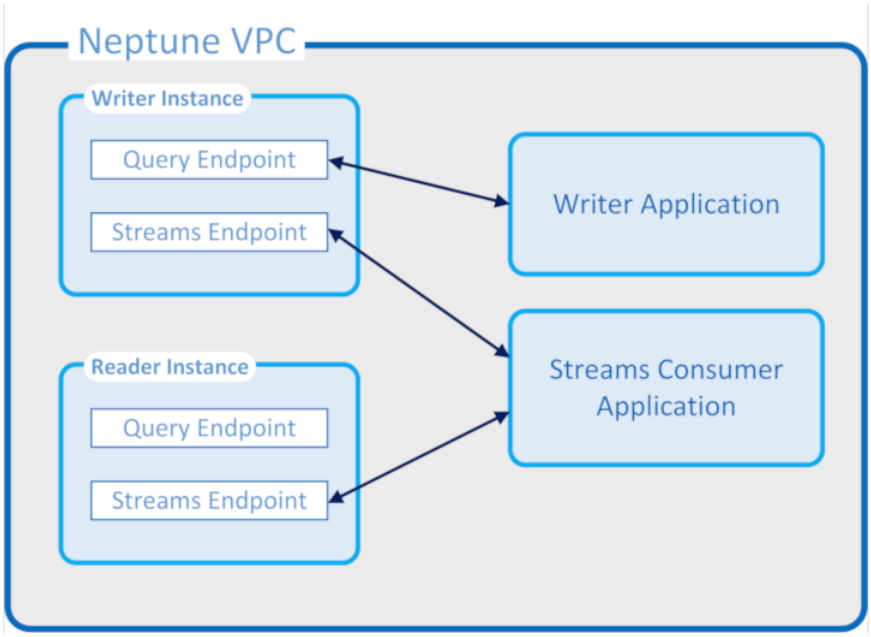
> **Note**
>
> Questa funzionalità è disponibile in modalità di laboratorio a partire dal Rilascio 1.0.1.0.200463.0 (15/10/2019), ed è disponibile per l'uso in produzione a partire dal rilascio 1.0.2.2.R2 del motore Neptune.

Di seguito sono elencati alcuni dei numerosi casi d'uso in cui è possibile acquisire automaticamente le modifiche apportate a un grafo:

- Potrebbe essere necessario che l'applicazione invii automaticamente una notifica alle persone quando vengono apportate determinate modifiche.
- Potresti voler mantenere una versione attuale dei tuoi dati grafici anche in un altro data store, come Amazon OpenSearch Service, Amazon o Amazon ElastiCache Simple Storage Service (Amazon S3).

Neptune utilizza per il flusso di log delle modifiche lo stesso spazio di archiviazione nativo dei dati del grafo. Scrive le voci change log in modo sincrono insieme alla transazione che apporta tali modifiche. Puoi recuperare questi record di modifica dal flusso di log tramite un'API REST HTTP. (Per informazioni, consulta Chiamata dell'API Streams.)

Il seguente diagramma mostra come recuperare i dati di log delle modifiche da Neptune Streams.



Garanzie di Neptune Streams

- Le modifiche apportate da una transazione sono immediatamente disponibili per la lettura da scrittori e lettori non appena la transazione viene completata (a parte il normale ritardo di replica nei lettori).
- I record di modifica vengono visualizzati rigorosamente in sequenza, nell'ordine in cui si sono verificati (incluse le modifiche apportate all'interno di una transazione).
- I flussi di modifiche non contengono duplicati. Ogni modifica viene registrata una sola volta.
- I flussi di modifiche sono completi. Nessuna modifica viene persa o omessa.
- I flussi di modifiche contengono tutte le informazioni necessarie per determinare lo stato completo del database stesso in qualsiasi momento, a condizione che lo stato di avvio sia noto.
- I flussi possono essere attivati o disattivati in qualsiasi momento.

Proprietà operative di Neptune Streams

- Il flusso di log delle modifiche è completamente gestito.
- I dati di log delle modifiche vengono scritti in modo sincrono come parte della stessa transazione che apporta una modifica.
- Quando la funzionalità Neptune Streams è abilitata, vengono addebitati i costi di I/O e archiviazione associati ai dati di log delle modifiche.

- Per impostazione predefinita, i record di modifica vengono eliminati automaticamente una settimana dopo la creazione. A partire dal rilascio 1.2.0.0 del motore, questo periodo di conservazione può essere modificato utilizzando il parametro del cluster database neptune_streams_expiry_days con un numero qualsiasi di giorni compreso tra 1 e 90.
- Le prestazioni di lettura sui flussi vengono calibrate con le istanze.
- Puoi ottenere disponibilità elevata e throughput di lettura utilizzando repliche di lettura. Non esiste un limite al numero di lettori di flussi che puoi creare e utilizzare simultaneamente.
- I dati di log delle modifiche vengono replicati su più zone di disponibilità, rendendoli estremamente durevoli.
- I dati di log sono protetti quanto i dati del grafo stessi. Possono essere crittografati mentre sono inattivi o in transito. L'accesso può essere controllato tramite IAM, Amazon VPC e AWS Key Management Service ()AWS KMS. Come i dati del grafico, è possibile eseguirne il backup e successivamente ripristinarli utilizzando i point-in-time ripristini (PITR).
- La scrittura sincrona dei dati di flusso come parte di ogni transazione causa un piccolo peggioramento delle prestazioni complessive di scrittura.
- I dati di flusso non vengono partizionati, perché Neptune è appositamente progettato per essere a partizione singola.
- L'API `GetRecords` del flusso di log utilizza le stesse risorse di tutte le altre operazioni del grafo Neptune. Ciò significa che i client devono eseguire il bilanciamento del carico tra le richieste di flusso e altre richieste database.
- Quando i flussi sono disabilitati, tutti i dati di log diventano immediatamente inaccessibili. Ciò significa che devi leggere tutti i dati di log che potrebbero interessarti prima di disabilitare la registrazione.
- Al momento non esiste un'integrazione nativa con. AWS Lambda Il flusso di log non genera un evento che può attivare una funzione Lambda.

Argomenti

- Utilizzo di Neptune Streams
- Formati di serializzazione in Neptune Streams
- Esempi di Neptune Streams
- Utilizzo AWS CloudFormation per configurare la replica da Neptune-Neptune con l'applicazione Streams Consumer
- Utilizzo della replica tra regioni di Neptune Streams per il ripristino di emergenza

Utilizzo di Neptune Streams

La funzionalità Neptune Streams consente di generare una sequenza completa di voci di log delle modifiche che registrano automaticamente e in tempo reale ogni modifica apportata ai dati del grafo. Per una panoramica di questa caratteristica, consulta Acquisizione di modifiche al grafo in tempo reale mediante Neptune Streams.

Argomenti

Abilitazione di Neptune Streams

È possibile abilitare o disabilitare Neptune Streams in qualsiasi momento impostando il parametro del cluster database `neptune_streams`. L'impostazione del parametro su `1` abilita i flussi, mentre l'impostazione su `0` disabilita i flussi.

> **Note**
>
> Dopo aver modificato il parametro del cluster database `neptune_streams`, è necessario riavviare tutte le istanze database nel cluster per rendere effettiva la modifica.

È possibile impostare il parametro del cluster database neptune_streams_expiry_days per controllare per quanti giorni, da 1 a 90, i record di flusso rimangono sul server prima di essere eliminati. Il valore predefinito è 7.

Neptune Streams è stato inizialmente introdotto come funzionalità sperimentale che era possibile abilitare o disabilitare in modalità di laboratorio utilizzando il parametro del cluster database `neptune_lab_mode` (vedi Modalità di laboratorio Neptune). L'utilizzo della modalità laboratorio per abilitare Streams è ora obsoleto e verrà disattivato in futuro.

Disabilitazione di Neptune Streams

Puoi disattivare Neptune Streams in qualsiasi momento in cui è in esecuzione.

Per disattivare Streams, aggiorna il gruppo di parametri Cluster DB in modo che il valore del parametro `neptune_streams` sia impostato su 0.

> **Important**
>
> Non appena Streams viene disattivato, non puoi più accedere ai dati di log delle modifiche. Assicurati di leggere ciò che ti interessa prima di disattivare Streams.

Chiamata di REST API di Neptune Streams

Puoi accedere a Neptune Streams utilizzando una REST API che invia una richiesta HTTP GET a uno dei seguenti endpoint locali:

- Per un database grafico SPARQL: `https://`*`Neptune-DNS`*`:8182/sparql/stream`.
- Per un database a grafo Gremlin o openCypher: `https://`*`Neptune-DNS`*`:8182/propertygraph/stream` o `https://`*`Neptune-DNS`*`:8182/pg/stream`.

> **Note**
>
> A partire dal rilascio 1.1.0.0 del motore, l'endpoint del flusso Gremlin (`https://`*`Neptune-DNS`*`:8182/gremlin/stream`) è diventato obsoleto, insieme al formato di output associato (`GREMLIN_JSON`). È ancora supportato per la compatibilità con le versioni precedenti, ma potrebbe essere rimosso nei rilasci futuri.

È consentita una sola operazione `GET` HTTP.

Neptune supporta la compressione `gzip` della risposta, purché la richiesta HTTP includa un'intestazione `Accept-Encoding` che specifichi `gzip` come formato di compressione accettato (ovvero `"Accept-Encoding: gzip"`).

Parametri

- `limit`: long, facoltativo. Intervallo: 1-100.000. Impostazione predefinita: 10

Specifica il numero massimo di record da restituire. Esiste anche un limite di dimensioni di 10 MB per la risposta che non può essere modificato e che ha la precedenza sul numero di record specificato nel parametro `limit`. La risposta include un record che supera la soglia se il limite di 10 MB è stato raggiunto.

- `iteratorType`: stringa, facoltativo.

  Questo parametro può accettare uno dei seguenti valori:

  - `AT_SEQUENCE_NUMBER`: (valore predefinito) indica che la lettura deve iniziare dal numero di sequenza di eventi specificato congiuntamente dai parametri `commitNum` e `opNum`.
  - `AFTER_SEQUENCE_NUMBER`: indica che la lettura deve iniziare immediatamente dopo il numero di sequenza di eventi specificato congiuntamente dai parametri `commitNum` e `opNum`.
  - `TRIM_HORIZON`: indica che la lettura deve iniziare dall'ultimo record non tagliato nel sistema, ovvero il record più vecchio non scaduto (non ancora eliminato) nel flusso di log delle modifiche. Questa modalità è utile durante l'avvio dell'applicazione, quando non si dispone di un numero di sequenza dell'evento di avvio specifico.
  - `LATEST`: indica che la lettura deve iniziare dal record più recente non tagliato nel sistema, ovvero il record più recente non scaduto (non ancora eliminato) nel flusso di log delle modifiche. Questa modalità è utile quando è necessario leggere i record dalla parte superiore del flusso per non elaborare i record più vecchi, ad esempio durante un ripristino di emergenza o un aggiornamento senza tempi di inattività. Notare che in questa modalità viene restituito al massimo un solo record.

- `commitNum`: long, obbligatorio quando iteratorType è `AT_SEQUENCE_NUMBER` o `AFTER_SEQUENCE_NUMBER`.

  Il numero di commit del record iniziale da leggere dal flusso change-log.

  Questo parametro viene ignorato quando `iteratorType` è `TRIM_HORIZON` o `LATEST`.

- `opNum`: long, facoltativo (il valore predefinito è `1`).

  Il numero di sequenza dell'operazione all'interno del commit specificato da cui iniziare la lettura nei dati del flusso change-log.

Le operazioni che modificano i dati del grafo SPARQL generano solitamente solo un singolo record di modifica per operazione. Tuttavia, le operazioni che modificano i dati del grafo Gremlin possono generare più record di modifica per operazione, come negli esempi seguenti:

- `INSERT`: un vertice Gremlin può avere più etichette e un elemento Gremlin può avere più proprietà. Un record di modifica separato viene generato per ogni etichetta e proprietà quando si inserisce un elemento.
- `UPDATE`: quando la proprietà di un elemento Gremlin viene modificata, vengono generati due record di modifica: il primo per la rimozione del valore precedente e il secondo per l'inserimento del nuovo valore.
- `DELETE`: viene generato un record di modifica separato per ogni proprietà dell'elemento che viene eliminata. Ad esempio, quando un edge Gremlin con proprietà viene eliminato, viene generato un record di modifica per ciascuna delle proprietà e successivamente ne viene generato uno per l'eliminazione dell'etichetta edge.

  Quando un vertice Gremlin viene eliminato, vengono eliminate per prime tutte le proprietà edge in entrata e in uscita, quindi le etichette edge, le proprietà dei vertici e infine le etichette dei vertici. Ognuna di queste eliminazioni genera un record di modifica.

Formato della risposta API di Neptune Streams

Una risposta a una richiesta REST API di Neptune Streams include i campi seguenti:

- `lastEventId`: identificatore di sequenza dell'ultima modifica nella risposta del flusso. Un ID evento è composto da due campi: un `commitNum` che identifica una transazione che ha modificato il grafo e un `opNum` che identifica un'operazione specifica all'interno di tale transazione. Questo viene mostrato nell'esempio seguente.

  ```
  "eventId": {
    "commitNum": 12,
    "opNum": 1
  }
  ```

- `lastTrxTimestamp`: l'ora in cui è stato richiesto il commit per la transazione, in millisecondi dall'epoca Unix.
- `format`: formato di serializzazione per i record di modifica restituiti. I valori possibili sono `PG_JSON` per i record di modifica Gremlin o openCypher e `NQUADS` per i record di modifica SPARQL.
- `records`: un array di record serializzati del flusso di log delle modifiche inclusi nella risposta. Ogni record dell'array `records` contiene i seguenti campi:
  - `commitTimestamp`: l'ora in cui è stato richiesto il commit per la transazione, in millisecondi dall'epoca Unix.

- `eventId`: identificatore di sequenza del record di modifica del flusso.
- `data`— Il record serializzato di Gremlin, SPARQL o change. OpenCypher I formati di serializzazione di ciascun record sono descritti più dettagliatamente nella sezione successiva, Formati di serializzazione in Neptune Streams.
- `op`: operazione che ha creato la modifica.
- `isLastOp`: presente solo se questa operazione è l'ultima della transazione. Se presente, è impostato su `true`. È utile per garantire che venga consumata un'intera transazione.

- `totalRecords`: numero totale di record nella risposta.

Ad esempio, la seguente risposta restituisce i dati di modifica di Gremlin, per una transazione che contiene più di un'operazione:

```
{
  "lastEventId": {
    "commitNum": 12,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1560011610678,
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "ADD"
    }
  ],
  "totalRecords": 1
```

```
}
```

La seguente risposta restituisce i dati di modifica SPARQL per l'ultima operazione di una transazione (operazione identificata da `EventId(97, 1)` nella transazione numero 97).

```
{
  "lastEventId": {
    "commitNum": 97,
    "opNum": 1
  },
  "lastTrxTimestamp": 1561489355102,
  "format": "NQUADS",
  "records": [
    {
      "commitTimestamp": 1561489355102,
      "eventId": {
        "commitNum": 97,
        "opNum": 1
      },
      "data": {
        "stmt": "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Eccezioni API di Neptune Streams

La tabella seguente descrive le eccezioni di Neptune Streams.

Codice di errore	Codice HTTP	OK riprovare?	Messaggio
`InvalidParameterException`	400	No	È stato fornito un out-of-range valore or non valido come parametro di input.

Codice di errore	Codice HTTP	OK riprovare?	Messaggio
`ExpiredStreamException`	400	No	Tutti i record richiesti superano l'età massima consentita e sono scaduti.
`ThrottlingException`	500	Sì	La velocità delle richieste supera il throughput massimo consentito.
`StreamRecordsNotFoundException`	404	No	Impossibile trovare la risorsa richiesta. Il flusso potrebbe non essere specificato correttamente.
`MemoryLimitExceededException`	500	Sì	L'elaborazione della richiesta non è andata a buon fine a causa della mancanza di memoria, tuttavia si potrà riprovare quando il server sarà meno occupato.

Formati di serializzazione in Neptune Streams

Amazon Neptune utilizza due diversi formati per la serializzazione dei dati delle modifiche al grafo in flussi di log, a seconda che il grafo sia stato creato utilizzando Gremlin o SPARQL.

Entrambi i formati condividono un formato di serializzazione dei record comune, come descritto in Formato della risposta API di Neptune Streams, che contiene i seguenti campi:

- `commitTimestamp`: l'ora in cui è stato richiesto il commit per la transazione, in millisecondi dall'epoca Unix.

- `eventId`: identificatore di sequenza del record di modifica del flusso.
- `data`— Il record serializzato Gremlin, SPARQL o Change. OpenCypher I formati di serializzazione di ciascun record sono descritti più dettagliatamente nelle sezioni successive.
- `op`: operazione che ha creato la modifica.

Argomenti

Formato di serializzazione delle modifiche PG_JSON

> **Note**
>
> A partire dal rilascio 1.1.0.0 del motore, il formato di output del flusso Gremlin (`GREMLIN_JSON`) generato dall'endpoint del flusso Gremlin (`https://Neptune-DNS:8182/gremlin/stream`) è diventato obsoleto. È stato sostituito da PG_JSON, che attualmente è identico a `GREMLIN_JSON`.

Un record di modifica Gremlin o openCypher, contenuto nel campo `data` di una risposta del flusso di log, dispone dei seguenti campi:

- `id`: stringa, obbligatorio.

  L'ID dell'elemento Gremlin o openCypher.
- `type`: stringa, obbligatorio.

  Il tipo di questo elemento Gremlin o openCypher. Deve essere uno dei seguenti:
  - `vl`: etichetta di vertice per Gremlin; etichetta di nodo per openCypher.
  - `vp`: proprietà di vertice per Gremlin; proprietà di nodo per openCypher.
  - `e`: arco ed etichetta di arco per Gremlin; relazione e tipo di relazione per openCypher.
  - `ep`: proprietà di arco per Gremlin; proprietà di relazione per openCypher.
- `key`: stringa, obbligatorio.

  Il nome della proprietà. Per le etichette degli elementi, questo è "label".

- `value`: oggetto `value`, obbligatorio.

  Si tratta di un oggetto JSON che contiene un campo `value` per il valore stesso e un campo `datatype` per il tipo di dati JSON di tale valore.

```
  "value": {
    "value": "the new value",
    "dataType": "the JSON datatype of the new value"
  }
```

- `from`: stringa, facoltativo.

  Se si tratta di un arco (tipo="e"), è l'ID del vertice from o del nodo di origine corrispondente.

- `to`: stringa, facoltativo.

  Se si tratta di un arco (tipo="e"), ID del vertice to corrispondente o del nodo di destinazione.

Esempi di Gremlin

- Di seguito è riportato un esempio di un'etichetta del vertice Gremlin.

```
{
  "id": "an ID string",
  "type": "vl",
  "key": "label",
  "value": {
    "value": "the new value of the vertex label",
    "dataType": "String"
  }
}
```

- Di seguito è riportato un esempio di una proprietà del vertice Gremlin.

```
{
  "id": "an ID string",
  "type": "vp",
  "key": "the property name",
  "value": {
    "value": "the new value of the vertex property",
    "dataType": "the datatype of the vertex property"
  }
```

```
}
```

- Di seguito è riportato un esempio di un edge Gremlin.

```
{
  "id": "an ID string",
  "type": "e",
  "key": "label",
  "value": {
    "value": "the new value of the edge",
    "dataType": "String"
  },
  "from": "the ID of the corresponding "from" vertex",
  "to": "the ID of the corresponding "to" vertex"
}
```

Esempi di openCypher

- Il seguente è un esempio di etichetta di nodo openCypher.

```
{
  "id": "an ID string",
  "type": "vl",
  "key": "label",
  "value": {
    "value": "the new value of the node label",
    "dataType": "String"
  }
}
```

- Il seguente è un esempio di proprietà di nodo openCypher.

```
{
  "id": "an ID string",
  "type": "vp",
  "key": "the property name",
  "value": {
    "value": "the new value of the node property",
    "dataType": "the datatype of the node property"
  }
}
```

- Il seguente è un esempio di relazione openCypher.

```
{
  "id": "an ID string",
  "type": "e",
  "key": "label",
  "value": {
    "value": "the new value of the relationship",
    "dataType": "String"
  },
  "from": "the ID of the corresponding source node",
  "to": "the ID of the corresponding target node"
}
```

Formato di serializzazione della modifica SPARQL NQUADS

Neptune registra le modifiche ai quad SPARQL nel grafo utilizzando il linguaggio `N-QUADS` di Resource Description Framework (RDF) definito nella specifica W3C RDF 1.1 N-Quads.

Il campo `data` nel record di modifica contiene semplicemente un campo `stmt` che contiene un'istruzione N-QUADS che esprime il quad modificato, come nell'esempio seguente.

```
  "stmt" : "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
```

Esempi di Neptune Streams

I seguenti esempi mostrano come accedere ai dati del flusso di log delle modifiche in Amazon Neptune.

Argomenti

- Change Log AT_SEQUENCE_NUMBER
- Change Log AFTER_SEQUENCE_NUMBER
- Change Log TRIM_HORIZON
- Change Log LATEST
- Log delle modifiche di compressione

Change Log **AT_SEQUENCE_NUMBER**

L'esempio seguente mostra un change log AT_SEQUENCE_NUMBER di Gremlin o openCypher.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AT_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
    {
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "commitTimestamp": 1560011610678,
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Questo esempio mostra un change log AT_SEQUENCE_NUMBER di SPARQL.

```
curl -s "https://localhost:8182/sparql/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AT_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
    "commitNum": 1,
```

```
    "opNum": 1
  },
  "lastTrxTimestamp": 1571252030566,
  "format": "NQUADS",
  "records": [
    {
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "commitTimestamp": 1571252030566,
      "data": {
        "stmt": "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Change Log **AFTER_SEQUENCE_NUMBER**

L'esempio seguente mostra un change log AFTER_SEQUENCE_NUMBER di Gremlin o openCypher.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AFTER_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
    "commitNum": 2,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011633768,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1560011633768,
      "eventId": {
        "commitNum": 2,
        "opNum": 1
      },
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
```

```
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "REMOVE",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Change Log **TRIM_HORIZON**

L'esempio seguente mostra un change log `TRIM_HORIZON` di Gremlin o openCypher.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&iteratorType=TRIM_HORIZON" |jq
{
  "lastEventId": {
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1560011610678,
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
```

```
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Change Log **LATEST**

L'esempio seguente mostra un change log LATEST di Gremlin o openCypher. Notare che i parametri API `limit`, `commitNum` e `opNum` sono completamente facoltativi.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?iteratorType=LATEST" | jq
{
  "lastEventId": {
    "commitNum": 21,
    "opNum": 4
  },
  "lastTrxTimestamp": 1634710497743,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1634710497743,
      "eventId": {
        "commitNum": 21,
        "opNum": 4
      },
      "data": {
        "id": "24be4e2b-53b9-b195-56ba-3f48fa2b60ac",
        "type": "e",
        "key": "label",
        "value": {
          "value": "created",
          "dataType": "String"
        },
        "from": "4",
        "to": "5"
      },
      "op": "REMOVE",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
```

```
}
```

Log delle modifiche di compressione

L'esempio seguente mostra un change log di compressione di Gremlin o openCypher.

```
curl -sH \
  "Accept-Encoding: gzip" \
  "https://Neptune-DNS:8182/propertygraph/stream?limit=1&commitNum=1" \
  -H "Accept-Encoding: gzip" \
  -v |gunzip -|jq
> GET /propertygraph/stream?limit=1 HTTP/1.1
> Host: localhost:8182
> User-Agent: curl/7.64.0
> Accept: /
> Accept-Encoding: gzip
*> Accept-Encoding: gzip*
>
< HTTP/1.1 200 OK
< Content-Type: application/json; charset=UTF-8
< Connection: keep-alive
*< content-encoding: gzip*
< content-length: 191
<
{ [191 bytes data]
Connection #0 to host localhost left intact
{
  "lastEventId": "1:1",
  "lastTrxTimestamp": 1558942160603,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1558942160603,
      "eventId": "1:1",
      "data": {
        "id": "v1",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "person",
          "dataType": "String"
        }
      },
```

```
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Utilizzo AWS CloudFormation per configurare la replica da Neptune-Neptune con l'applicazione Streams Consumer

È possibile utilizzare un AWS CloudFormation modello per configurare l'applicazione consumer Neptune Streams per supportare la replica da Neptune a Neptune.

Argomenti

- Scegli un modello per la tua regione AWS CloudFormation
- Aggiunta di dettagli relativi allo stack consumer di flussi Neptune che verranno creati
- Esegui il modello AWS CloudFormation
- Per aggiornare lo strumento per il polling dei flussi con gli ultimi artefatti Lambda

Scegli un modello per la tua regione AWS CloudFormation

Per avviare lo AWS CloudFormation stack appropriato sulla AWS CloudFormation console, scegli uno dei pulsanti Launch Stack nella tabella seguente, a seconda della AWS regione che desideri utilizzare.

Regione	Vista	Visualizzazione in Designer	Avvia
Stati Uniti orientali (Virginia settentrionale)	Visualizzazione	Visualizzazione in Designer	Launch Stack
Stati Uniti orientali (Ohio)	Visualizzazione	Visualizzazione in Designer	Launch Stack

Regione	Vista	Visualizzazione in Designer	Avvia
Stati Uniti occidentali (California settentrionale)	Visualizzazione	Visualizzazione in Designer	
US West (Oregon)	Visualizzazione	Visualizzazione in Designer	
Canada (Centrale)	Visualizzazione	Visualizzazione in Designer	
Sud America (San Paolo)	Visualizzazione	Visualizzazione in Designer	
Europa (Stoccolma)	Visualizzazione	Visualizzazione in Designer	
Europa (Irlanda)	Visualizzazione	Visualizzazione in Designer	
Europa (Londra)	Visualizzazione	Visualizzazione in Designer	
Europa (Parigi)	Visualizzazione	Visualizzazione in Designer	
Europa (Francoforte)	Visualizzazione	Visualizzazione in Designer	
Medio Oriente (Bahrein)	Visualizzazione	Visualizzazione in Designer	
Medio Oriente (Emirati Arabi Uniti)	Visualizzazione	Visualizzazione in Designer	
Israele (Tel Aviv)	Visualizzazione	Visualizzazione in Designer	

Regione	Vista	Visualizzazione in Designer	Avvia
Africa (Città del Capo)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Tokyo)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Hong Kong)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Seul)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Singapore)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Sydney)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Mumbai)	Visualizzazione	Visualizzazione in Designer	
Cina (Pechino)	Visualizzazione	Visualizzazione in Designer	
Cina (Ningxia)	Visualizzazione	Visualizzazione in Designer	
AWS GovCloud (Stati Uniti occidentali)	Visualizzazione	Visualizzazione in Designer	
AWS GovCloud (Stati Uniti orientali)	Visualizzazione	Visualizzazione in Designer	

Nella pagina Create Stack (Crea stack) scegliere Next (Avanti).

Aggiunta di dettagli relativi allo stack consumer di flussi Neptune che verranno creati

La pagina Specify Stack Details (Specifica dettagli stack) fornisce proprietà e parametri che è possibile utilizzare per controllare la configurazione dell'applicazione:

Nome dello stack: il nome del nuovo AWS CloudFormation stack che stai creando. In genere è possibile utilizzare il valore predefinito `NeptuneStreamPoller`.

In Parametri, fornire quanto segue:

Configurazione di rete per il VPC in cui viene eseguita l'applicazione consumer di flussi

- **`VPC`**: specifica il nome del VPC in cui verrà eseguita la funzione Lambda di polling.
- **`SubnetIDs`**: sottoreti in cui viene stabilita un'interfaccia di rete. Aggiungere le sottoreti corrispondenti al cluster Neptune.
- **`SecurityGroupIds`**: specifica gli ID dei gruppi di sicurezza che concedono l'accesso in ingresso in scrittura al cluster database Neptune di origine.
- **`RouteTableIds`**: necessario per creare un endpoint Amazon DynamoDB nel VPC Neptune, se non è presente. È necessario fornire un elenco separato da virgole di ID tabella di routing associati alle sottoreti.
- **`CreateDDBVPCEndPoint`**: valore booleano il cui valore predefinito è `true`, che indica se è necessario creare o meno un endpoint VPC Dynamo DB. È necessario modificarlo e impostarlo su `false` solo se è già stato creato un endpoint DynamoDB nel VPC.
- **`CreateMonitoringEndPoint`**: valore booleano il cui valore predefinito è `true`, che indica se è necessario creare o meno un endpoint VPC di monitoraggio. È necessario modificarlo in `false` solo se è già stato creato un endpoint di monitoraggio nel VPC.

Polling di flussi

- **`ApplicationName`**: in genere è possibile lasciare questo valore impostato sul valore predefinito (`NeptuneStream`). Se si utilizza un nome diverso, deve essere univoco.
- **`LambdaMemorySize`**: consente di impostare le dimensioni della memoria disponibili per la funzione Lambda dello strumento di polling. Il valore predefinito è 2.048 megabyte.
- **`LambdaRuntime`**: linguaggio utilizzato nella funzione Lambda che recupera gli elementi dal flusso Neptune. Può essere impostato su `python3.9` o su `java8`.

- **`LambdaS3Bucket`**: bucket Amazon S3 che contiene artefatti del codice Lambda. Lasciare questo parametro vuoto, a meno che non si utilizzi una funzione di polling Lambda personalizzata che viene caricata da un bucket Amazon S3 diverso.
- **`LambdaS3Key`**: chiave Amazon S3 che corrisponde agli artefatti del codice Lambda. Lasciare questo parametro vuoto, a meno che non si utilizzi una funzione di polling Lambda personalizzata.
- **`LambdaLoggingLevel`**: in generale, lasciare il valore predefinito, `INFO`.
- **`ManagedPolicies`**: elenca le policy gestite da utilizzare per l'esecuzione della funzione Lambda. In generale, lasciare vuoto a meno che non si utilizzi una funzione di polling Lambda personalizzata.
- **`StreamRecordsHandler`**: in generale, lascia vuoto a meno che non utilizzi un gestore personalizzato per i record nei flussi Neptune.
- **`StreamRecordsBatchSize`**: numero massimo di registri da recuperare dal flusso. È possibile utilizzare questo parametro per ottimizzare le prestazioni. Il valore predefinito (`5000`) è un buon punto di partenza. Il massimo consentito è 10.000. Più alto è il numero, meno chiamate di rete sono necessarie per leggere i record dal flusso, ma maggiore è la memoria necessaria per elaborare i record. Valori più bassi di questo parametro comportano una minore velocità di trasmissione effettiva.
- **`MaxPollingWaitTime`**: tempo di attesa massimo tra due polling (in secondi). Determina la frequenza con cui viene richiamato lo strumento di polling Lambda per il polling dei flussi Neptune. Impostare questo valore su 0 per il polling continuo. Il valore massimo è di 3.600 secondi (1 ora). Il valore predefinito (60 secondi) è un buon punto di partenza, a seconda della velocità con cui cambiano i dati del grafico.
- **`MaxPollingInterval`**: periodo massimo di polling continuo (in secondi). Consente di impostare un timeout per la funzione di polling Lambda. Il valore deve essere compreso tra 5 e 900 secondi. Il valore predefinito (600 secondi) è un buon punto di partenza.
- **`StepFunctionFallbackPeriod`**— Il numero di unità di cui step-function-fallback-period attendere il poller, dopodiché la funzione step viene richiamata tramite Amazon CloudWatch Events per il ripristino in caso di errore. Il valore predefinito (5 minuti) è un buon punto di partenza.
- **`StepFunctionFallbackPeriodUnit`**: unità di tempo utilizzate per misurare il periodo `StepFunctionFallbackPeriodUnit` precedente (`minutes`, `hours` o `days`). Il valore predefinito (`minutes`) è generalmente sufficiente.

Flusso Neptune

- **`NeptuneStreamEndpoint`**: (obbligatorio) endpoint del flusso di origine Neptune. Può assumere uno dei due formati seguenti:
  - **`https://`*`your DB cluster`*`:`*`port`*`/propertygraph/stream`** (o il relativo alias, `https://`*`your DB cluster`*`:`*`port`*`/pg/stream`).
  - **`https://`*`your DB cluster`*`:`*`port`*`/sparql/stream`**.
- **`Neptune Query Engine`**: scegliere Gremlin, openCypher o SPARQL.
- **`IAMAuthEnabledOnSourceStream`**: se il cluster database Neptune utilizza l'autenticazione IAM, impostare questo parametro su `true`.
- **`StreamDBClusterResourceId`**: se il cluster database Neptune utilizza l'autenticazione IAM, impostare questo parametro sull'ID risorsa del cluster. L'ID risorsa non è lo stesso dell'ID cluster. Invece, assume il formato: `cluster-` seguito da 28 caratteri alfanumerici. Può essere trovato in Dettagli del cluster nella console Neptune.

Cluster database Neptune di destinazione

- **`TargetNeptuneClusterEndpoint`**: endpoint (solo nome host) del cluster di backup di destinazione.

  Notare che se si specifica `TargetNeptuneClusterEndpoint`, non si può specificare `TargetSPARQLUpdateEndpoint`.
- **`TargetNeptuneClusterPort`**: numero di porta per il cluster di destinazione.

  Notare che se si specifica `TargetSPARQLUpdateEndpoint`, l'impostazione di `TargetNeptuneClusterPort` viene ignorata.
- **`IAMAuthEnabledOnTargetCluster`**: impostare su true se l'autenticazione IAM deve essere abilitata sul cluster di destinazione.
- **`TargetAWSRegion`**— La AWS regione del cluster di backup di destinazione, ad esempio`us-east-1`). È necessario fornire questo parametro solo quando la AWS regione del cluster di backup di destinazione è diversa dalla regione del cluster di origine Neptune, come nel caso della replica tra regioni. Se le regioni di origine e di destinazione coincidono, questo parametro è facoltativo.

  Nota che se il `TargetAWSRegion` valore non è una AWS regione valida supportata da Neptune, il processo fallisce.

- **`TargetNeptuneDBClusterResourceId`**: (facoltativo) è necessario solo quando l'autenticazione IAM è abilitata nel cluster database di destinazione. Impostare sull'ID risorsa del cluster di destinazione.
- **`SPARQLTripleOnlyMode`**: flag booleano che determina se è abilitata la modalità solo tripla. In modalità solo tripla, non è disponibile la replica del grafo nominato. Il valore predefinito è `false`.
- **`TargetSPARQLUpdateEndpoint`**: URL dell'endpoint di destinazione per l'aggiornamento SPARQL, ad esempio `https://abc.com/xyz`. Questo endpoint può essere qualsiasi archivio SPARQL che supporti quadruple o triple.

  Notare che se si specifica `TargetSPARQLUpdateEndpoint`, non è possibile specificare anche `TargetNeptuneClusterEndpoint` e l'impostazione di `TargetNeptuneClusterPort` viene ignorata.
- **`BlockSparqlReplicationOnBlankNode`** — Bandiera booleana che, se impostata su`true`, interrompe la replica dei dati BlankNode in SPARQL (RDF). Il valore predefinito è `false`.

Allarme

- **`Required to create Cloud watch Alarm`**— Impostalo su `true` se vuoi creare un CloudWatch allarme per il nuovo stack.
- **`SNS Topic ARN for Cloudwatch Alarm Notifications`**— L'argomento SNS ARN in CloudWatch cui devono essere inviate le notifiche di allarme (necessario solo se gli allarmi sono abilitati).
- **`Email for Alarm Notifications`**: indirizzo e-mail a cui devono essere inviate le notifiche di allarme (necessario solo se gli allarmi sono abilitati).

Per la destinazione della notifica di allarme, è possibile aggiungere solo SNS, solo e-mail o sia SNS che e-mail.

Esegui il modello AWS CloudFormation

A questo punto è possibile completare il processo di provisioning di un'istanza dell'applicazione consumer di flussi Neptune come indicato di seguito:

1. Nella AWS CloudFormation pagina Specificare i dettagli dello stack, scegli Avanti.
2. Nella pagina Opzioni, scegli Avanti.

3. Nella pagina Revisione, seleziona la prima casella di controllo per accettare la creazione delle risorse IAM da parte di AWS CloudFormation . Seleziona la seconda casella di controllo per confermare `CAPABILITY_AUTO_EXPAND` per il nuovo stack.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` conferma in modo esplicito che, durante la creazione dello stack, le macro verranno ampliate senza revisione preventiva. Gli utenti spesso creano un set di modifiche da un modello elaborato, quindi le modifiche apportate dalle macro possono essere riesaminate prima dell'effettiva creazione dello stack. Per ulteriori informazioni, consulta l' AWS CloudFormation CreateStackAPI nell'AWS CloudFormation API Reference.

   Quindi, scegli Crea.

Per aggiornare lo strumento per il polling dei flussi con gli ultimi artefatti Lambda

Per aggiornare lo strumento per il polling dei flussi con gli ultimi artefatti Lambda, segui questa procedura:

1. In AWS Management Console, accedi AWS CloudFormation e seleziona lo AWS CloudFormation stack principale.
2. Seleziona l'opzione Aggiorna per lo stack.
3. Seleziona Sostituisci modello corrente.
4. Per l'origine del modello, scegli URL di Amazon S3 e immetti l'URL S3 seguente:

   ```
   https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
   neptune_to_neptune.json
   ```

5. Seleziona Avanti senza modificare alcun AWS CloudFormation parametro.
6. Scegliere Update Stack (Aggiorna stack).

Lo stack ora aggiornerà gli artefatti Lambda con quelli più recenti.

Utilizzo della replica tra regioni di Neptune Streams per il ripristino di emergenza

Neptune offre due modi per implementare le funzionalità di failover tra regioni:

- Copia e ripristino di snapshot tra regioni
- Utilizzo di Neptune Streams per replicare i dati tra due cluster in due regioni diverse.

La copia e il ripristino di snapshot tra regioni presentano il sovraccarico operativo più basso per il ripristino di un cluster Neptune in una regione diversa. Tuttavia, la copia di uno snapshot tra regioni può richiedere tempi di trasferimento dei dati significativi, poiché uno snapshot è un backup completo del cluster Neptune. Di conseguenza, la copia e il ripristino di snapshot tra regioni possono essere utilizzati per scenari che richiedono solo un obiettivo del punto di ripristino (RPO) di ore e un obiettivo del tempo di ripristino (RTO) di ore.

L'obiettivo del punto di ripristino (RPO) viene misurato in base al tempo che intercorre tra i backup. Definisce la quantità di dati che possono andare persi tra il momento in cui è stato eseguito l'ultimo backup e il momento in cui il database viene ripristinato.

L'obiettivo del tempo di ripristino (RTO) viene misurato in base al tempo necessario per eseguire un'operazione di ripristino. Questo è il tempo impiegato dal cluster database per eseguire il failover su un database ripristinato dopo che si è verificato un errore.

Neptune Streams offre un modo per mantenere sempre sincronizzato un cluster Neptune di backup con il cluster di produzione primario. Se si verifica un errore, il database esegue il failover sul cluster di backup. Ciò riduce gli obiettivi RPO e RTO a pochi minuti, poiché i dati vengono costantemente copiati nel cluster di backup, che è immediatamente disponibile come destinazione di failover in qualsiasi momento.

Lo svantaggio dell'utilizzo di Neptune Streams in questo modo è che sia il sovraccarico operativo necessario per mantenere i componenti di replica, sia il costo di avere sempre un secondo cluster database Neptune online possono essere significativi.

Configurazione della replica Neptune-Neptune

Il cluster database di produzione primario risiede in un VPC di una determinata regione di origine. Sono tre gli elementi principali che è necessario replicare o emulare in una regione di ripristino diversa ai fini del ripristino di emergenza:

- I dati archiviati nel cluster.
- La configurazione del cluster primario. Ad esempio, se utilizza l'autenticazione IAM, se è crittografato, i parametri del cluster database, i parametri delle istanze, le dimensioni delle istanze e così via.
- La topologia di rete utilizzata, inclusi il VPC di destinazione, i relativi gruppi di sicurezza e così via.

Per raccogliere queste informazioni è possibile utilizzare le API di gestione Neptune, come le seguenti:

- `DescribeDBClusters`
- `DescribeDBInstances`
- `DescribeDBClusterParameters`
- `DescribeDBParameters`
- `DescribeVpcs`

Con le informazioni raccolte, è possibile utilizzare la seguente procedura per configurare un cluster di backup in una regione diversa, sul quale il cluster di produzione può eseguire il failover in caso di errore.

1: Abilitare Neptune Streams

È possibile usare ModifyDBClusterParameterGroup per impostare il parametro `neptune_streams` su 1. Quindi, riavviare tutte le istanze del cluster database in modo che la modifica abbia effetto.

È consigliabile eseguire almeno un'operazione di aggiunta o aggiornamento sul cluster database di origine dopo aver abilitato Neptune Streams. In questo modo il flusso di modifiche viene popolato con punti dati a cui è possibile fare riferimento in seguito quando si risincronizza il cluster di produzione con il cluster di backup.

2: Creare un nuovo VPC nella regione in cui si desidera configurare il cluster di backup

Prima di creare un nuovo cluster database Neptune in una regione diversa dal cluster primario, è necessario stabilire un nuovo VPC nella regione di destinazione per ospitare il cluster. La connettività tra il cluster primario e quello di backup viene stabilita tramite il peering VPC, che utilizza il traffico attraverso sottoreti private in diversi VPC. Tuttavia, per stabilire il peering VPC tra due VPC, questi non devono avere blocchi CIDR o spazi di indirizzi IP sovrapposti. Ciò significa che non è possibile

usare il VPC predefinito in entrambe le regioni, perché il blocco CIDR per un VPC predefinito è sempre lo stesso (`172.31.0.0/16`).

È possibile utilizzare un VPC esistente nella regione di destinazione, purché soddisfi le seguenti condizioni:

- Non deve avere un blocco CIDR che si sovrappone al blocco CIDR del VPC in cui si trova il cluster primario.
- Non deve essere già connesso tramite peering con un altro VPC con lo stesso blocco CIDR del VPC in cui si trova il cluster primario.

Se non è disponibile un VPC adatto nella regione di destinazione, creane uno utilizzando l'API `CreateVpc` di Amazon EC2.

3: Creare uno snapshot del cluster primario e ripristinarlo nella regione di backup di destinazione

Creare ora un nuovo cluster Neptune in un VPC appropriato nella regione di backup di destinazione che è una copia del cluster di produzione:

Creare una copia del cluster di produzione nella regione di backup

1. Nella regione di backup di destinazione, ricreare i parametri e i gruppi di parametri utilizzati dal cluster database di produzione. A tale scopo, è possibile utilizzare `CreateDBClusterParameterGroup`, `CreateDBParameterGroup`, `ModifyDBClusterParameterGroup` e `ModifyDBParameterGroup`.

   Notare che le API `CopyDBClusterParameterGroup` e `CopyDBParameterGroup` non supportano attualmente la copia tra regioni.
2. Utilizzare `CreateDBClusterSnapshot` per creare uno snapshot del cluster di produzione nel VPC della regione di produzione.
3. Utilizzare `CopyDBClusterSnapshot` per copiare lo snapshot nel VPC della regione di backup di destinazione.
4. Utilizzare `RestoreDBClusterFromSnapshot` per creare un nuovo cluster database nel VPC della regione di backup di destinazione utilizzando lo snapshot copiato. Utilizzare le impostazioni e i parametri di configurazione copiati dal cluster di produzione primario.

5. Il nuovo cluster Neptune ora esiste ma non contiene istanze. Utilizzare `CreateDBInstance` per creare una nuova istanza primaria/di scrittura con lo stesso tipo e le stesse dimensioni dell'istanza di scrittura del cluster di produzione. A questo punto non è necessario creare repliche di lettura aggiuntive, a meno che l'istanza di backup non venga utilizzata per gestire l'I/O di lettura nella regione di destinazione prima di un failover.

4: Stabilire il peering VPC tra il VPC del cluster primario e il VPC del nuovo cluster di backup

Configurando il peering VPC, si consente al VPC del cluster primario di comunicare con il VPC del cluster di backup come se si trattasse di un'unica rete privata. Per questa opzione, effettua la procedura riportata di seguito.

1. Dal VPC del cluster di produzione, chiamare l'API `CreateVpcPeeringConnection` per stabilire la connessione peering.
2. Dal VPC del cluster di backup di destinazione, chiamare l'API `AcceptVpcPeeringConnection` per accettare la connessione peering.
3. Dal VPC del cluster di produzione, utilizzare l'API `CreateRoute` per aggiungere una route alla tabella di routing del VPC che reindirizza tutto il traffico al blocco CIDR del VPC di destinazione in modo che utilizzi l'elenco dei prefissi di peering VPC.
4. Analogamente, dal VPC del cluster di backup di destinazione, utilizzare l'API `CreateRoute` per aggiungere una route alla tabella di routing del VPC che instrada il traffico al VPC del cluster primario.

5: Configurare l'infrastruttura di replica di Neptune Streams

Ora che entrambi i cluster sono stati implementati e la comunicazione di rete tra le due regioni è stata stabilita, utilizza il modello Neptune-to-Neptune AWS CloudFormation per implementare la funzione Lambda consumer Neptune Streams con l'infrastruttura aggiuntiva che supporta la replica dei dati. Eseguire questa operazione nel VPC del cluster di produzione primario.

I parametri AWS CloudFormation che dovrai fornire per questo stack sono:

- **`NeptuneStreamEndpoint`**: endpoint del flusso per il cluster primario, in formato URL. Ad esempio: `https://(cluster name):8182/pg/stream`.
- **`QueryEngine`**: deve essere `gremlin`, `sparql` o `openCypher`.

- **`RouteTableIds`**: consente di aggiungere route sia per un endpoint VPC DynamoDB che per un endpoint VPC di monitoraggio.

  Anche i due parametri aggiuntivi, vale a dire `CreateMonitoringEndpoint` e `CreateDynamoDBEndpoint`, devono essere impostati su true se non sono già presenti nel VPC del cluster primario. Se esistono già, assicurati che siano impostati su false o la AWS CloudFormation creazione avrà esito negativo.
- **`SecurityGroupIds`**: specifica il gruppo di sicurezza utilizzato dal consumer Lambda per comunicare con l'endpoint del flusso Neptune del cluster primario.

  Nel cluster di backup di destinazione, collegare un gruppo di sicurezza che consenta il traffico proveniente da questo gruppo di sicurezza.
- **`SubnetIds`**: elenco di ID di sottorete nel VPC del cluster primario che possono essere utilizzati dal consumer Lambda per comunicare con il cluster primario.
- **`TargetNeptuneClusterEndpoint`**: endpoint (solo nome host) del cluster di backup di destinazione.
- **`TargetAWSRegion`**— La AWS regione del cluster di backup di destinazione, ad esempio`us-east-1`). È necessario fornire questo parametro solo quando la AWS regione del cluster di backup di destinazione è diversa dalla regione del cluster di origine Neptune, come nel caso della replica tra regioni. Se le regioni di origine e di destinazione coincidono, questo parametro è facoltativo.

  Nota che se il `TargetAWSRegion` valore non è una AWS regione valida supportata da Neptune, il processo fallisce.
- **`VPC`**: ID del VPC del cluster primario.

Tutti gli altri parametri possono essere lasciati con i valori predefiniti.

Una volta distribuito il AWS CloudFormation modello, Neptune inizierà a replicare tutte le modifiche dal cluster primario al cluster di backup. È possibile monitorare questa replica nei CloudWatch log generati dalla funzione consumer Lambda.

Altre considerazioni

- Se è necessario utilizzare l'autenticazione IAM tra il cluster primario e il cluster di backup, è possibile configurarla anche quando si richiama il modello. AWS CloudFormation

- Se nel cluster primario è abilitata la crittografia dei dati inattivi, considerare come gestire le chiavi KMS associate quando si copia lo snapshot nella regione di destinazione e come associare una nuova chiave KMS nella regione di destinazione.
- Una procedura consigliata consiste nell'utilizzare i CNAME DNS davanti agli endpoint Neptune utilizzati nelle applicazioni. Quindi, se è necessario eseguire manualmente il failover sul cluster di backup di destinazione, questi CNAME possono essere modificati in modo che puntino al cluster di destinazione e/o agli endpoint dell'istanza.

Ricerca nel testo completo in Amazon Neptune utilizzando Amazon Service OpenSearch

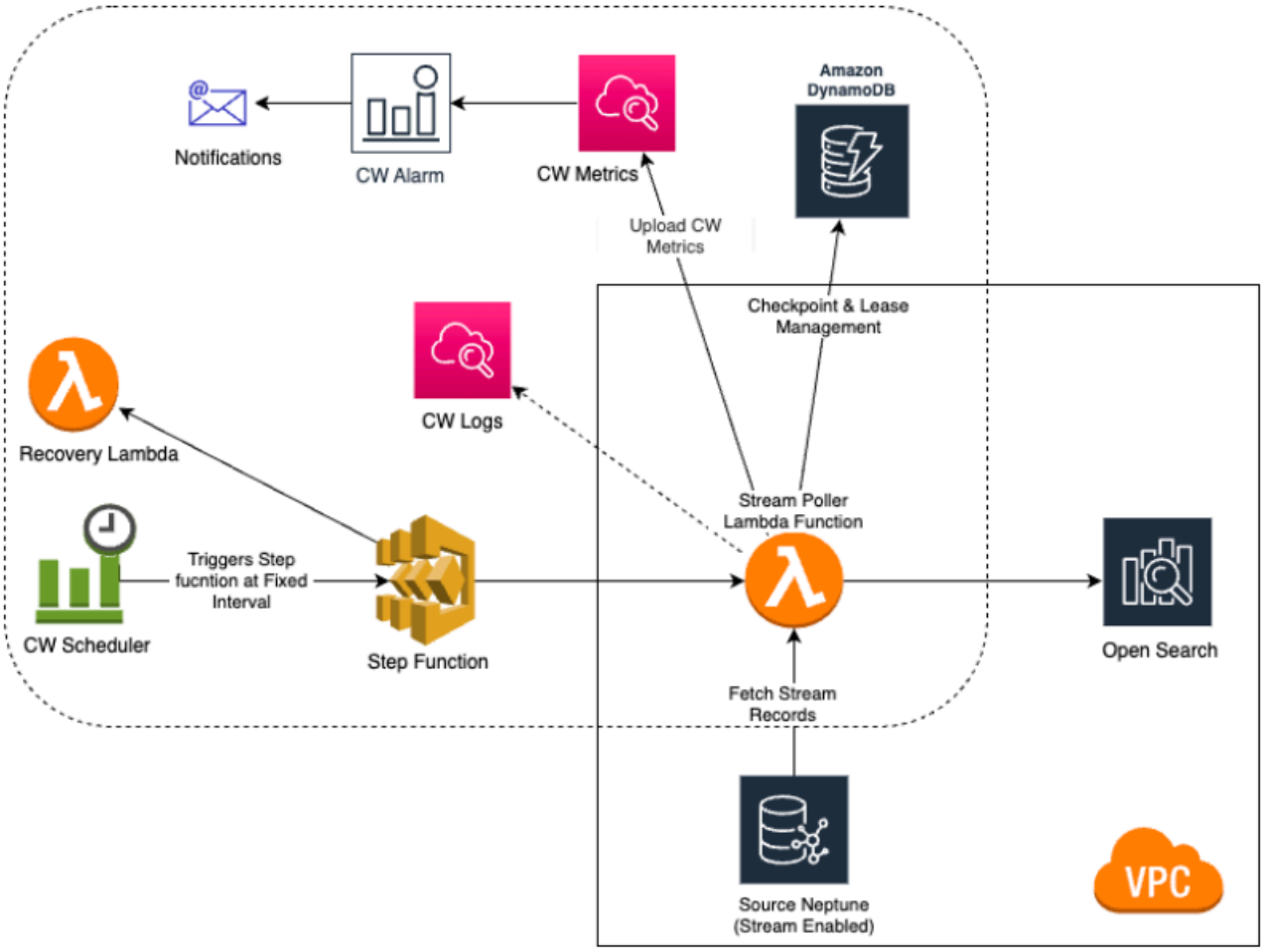
Neptune si integra con OpenSearch Amazon Service OpenSearch (Service) per supportare la ricerca di testo completo nelle query Gremlin e SPARQL. Questa funzionalità è disponibile a partire dal rilascio 1.0.2.1 del motore Neptune, anche se è consigliabile utilizzarla con il rilascio del motore `1.0.4.2` o successivo per sfruttare le correzioni più recenti.

A partire dalla versione 1.3.0.0 del motore, Amazon Neptune supporta l'utilizzo di Amazon OpenSearch Service Serverless per la ricerca di testo completo nelle query Gremlin e SPARQL.

> **Note**
>
> Durante l'integrazione con Amazon OpenSearch Service, Neptune richiede Elasticsearch versione 7.1 o successiva e funziona con 2.3, 2.5 e versioni successive. OpenSearch Neptune funziona anche con Serverless. OpenSearch

È possibile utilizzare Neptune con un cluster di servizi OpenSearch esistente che è stato popolato in base a. Modello di dati Neptune per i dati di OpenSearch In alternativa, puoi creare un dominio OpenSearch di servizio collegato a Neptune utilizzando uno stack. AWS CloudFormation

> **Important**
>
> Il processo di replica OpenSearch da Neptune descritto qui non replica i nodi vuoti. Si tratta di una limitazione importante da considerare.
> Inoltre, se abiliti il controllo granulare degli accessi sul tuo OpenSearch cluster, devi abilitare anche l'autenticazione IAM nel tuo database Neptune.



Argomenti

- Da Amazon Neptune a replica OpenSearch
- Replica in OpenSearch Serverless
- Esecuzione di query da un cluster OpenSearch con controllo granulare degli accessi attivato
- Utilizzo della sintassi di query Apache Lucene nelle query di ricerca full-text di Neptune
- Modello di dati Neptune per i dati di OpenSearch
- Parametri per la ricerca full-text Neptune
- Indicizzazione OpenSearch non tipo stringa in Amazon Neptune.
- Esecuzione di query di ricerca full-text in Amazon Neptune
- Query SPARQL di esempio che utilizzano la ricerca full-text in Neptune
- Utilizzo della ricerca full-text di Neptune nelle query Gremlin
- Risoluzione dei problemi di ricerca full-text di Neptune

Da Amazon Neptune a replica OpenSearch

Amazon Neptune supporta la ricerca di testo completo nelle query Gremlin e SPARQL utilizzando Amazon Service (Service). OpenSearch OpenSearch Puoi usare uno AWS CloudFormation stack per collegare un dominio di OpenSearch servizio a Neptune. Il AWS CloudFormation modello crea un'istanza applicativa streams-consumer che fornisce la replica da Neptune. OpenSearch

Prima di iniziare, è necessario un cluster Neptune DB esistente con stream abilitati per fungere da origine e OpenSearch un dominio di servizio che funga da destinazione di replica.

Se disponi già di un dominio di OpenSearch servizio di destinazione esistente a cui è possibile accedere tramite Lambda nel VPC in cui si trova il cluster Neptune DB, il modello può utilizzare quello. Altrimenti, è necessario crearne uno nuovo.

> **Note**
>
> Il OpenSearch cluster e la funzione Lambda che crei devono trovarsi nello stesso VPC del cluster Neptune DB e il cluster OpenSearch deve essere configurato in modalità VPC (non in modalità Internet).

Ti consigliamo di utilizzare un'istanza Neptune appena creata da utilizzare con Service. OpenSearch Se si utilizza un'istanza esistente che contiene già dei dati, è necessario eseguire una sincronizzazione dei dati del OpenSearch Servizio prima di effettuare le interrogazioni, altrimenti potrebbero esserci delle incongruenze nei dati. Questo GitHub progetto fornisce un esempio di come eseguire la sincronizzazione: Export Neptune OpenSearch to (https://github.com/awslabs/ /tree/ master/). amazon-neptune-tools export-neptune-to-elasticsearch

> **Important**
>
> Durante l'integrazione con Amazon OpenSearch Service, Neptune richiede la versione 7.1 o successiva di Elasticsearch e funziona con le versioni OpenSearch 2.3, 2.5 e future compatibili di Opensearch.

> **Note**
>
> A partire dalla versione 1.3.0.0 del motore, Amazon Neptune supporta l'utilizzo di Amazon OpenSearch Service Serverless per la ricerca di testo completo nelle query Gremlin e SPARQL.

Argomenti

Utilizzo di un AWS CloudFormation modello per avviare la replica di Neptune-to-Replica OpenSearch

Avvia uno AWS CloudFormation stack specifico per la tua regione

Ciascuno dei AWS CloudFormation modelli seguenti crea un'istanza di applicazione streams-consumer in una regione specifica AWS . Per avviare lo stack corrispondente utilizzando la AWS CloudFormation console, scegliete uno dei pulsanti Launch Stack nella tabella seguente, a seconda della AWS regione che desiderate utilizzare.

Regione	Vista	Visualizzazione in Designer	Avvia
Stati Uniti orientali (Virginia settentrionale)	Visualizzazione	Visualizzazione in Designer	
Stati Uniti orientali (Ohio)	Visualizzazione	Visualizzazione in Designer	

Regione	Vista	Visualizzazione in Designer	Avvia
Stati Uniti occidentali (California settentrionale)	Visualizzazione	Visualizzazione in Designer	
US West (Oregon)	Visualizzazione	Visualizzazione in Designer	
Canada (Centrale)	Visualizzazione	Visualizzazione in Designer	
Sud America (San Paolo)	Visualizzazione	Visualizzazione in Designer	
Europa (Stoccolma)	Visualizzazione	Visualizzazione in Designer	
Europa (Irlanda)	Visualizzazione	Visualizzazione in Designer	
Europa (Londra)	Visualizzazione	Visualizzazione in Designer	
Europa (Parigi)	Visualizzazione	Visualizzazione in Designer	
Europa (Francoforte)	Visualizzazione	Visualizzazione in Designer	
Medio Oriente (Bahrein)	Visualizzazione	Visualizzazione in Designer	
Medio Oriente (Emirati Arabi Uniti)	Visualizzazione	Visualizzazione in Designer	
Israele (Tel Aviv)	Visualizzazione	Visualizzazione in Designer	

Regione	Vista	Visualizzazione in Designer	Avvia
Africa (Città del Capo)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Hong Kong)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Tokyo)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Seul)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Singapore)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Mumbai)	Visualizzazione	Visualizzazione in Designer	
Cina (Pechino)	Visualizzazione	Visualizzazione in Designer	
Cina (Ningxia)	Visualizzazione	Visualizzazione in Designer	
AWS GovCloud (Stati Uniti occidentali)	Visualizzazione	Visualizzazione in Designer	
AWS GovCloud (Stati Uniti orientali)	Visualizzazione	Visualizzazione in Designer	

Nella pagina Create Stack (Crea stack) scegliere Next (Avanti).

Aggiungi dettagli sul nuovo OpenSearch stack che stai creando

La pagina Specifica i dettagli dello stack fornisce proprietà e parametri che puoi utilizzare per controllare la configurazione della ricerca full-text:

Nome dello stack: il nome del nuovo AWS CloudFormation stack che stai creando. In genere è possibile utilizzare il valore predefinito `NeptuneStreamPoller`.

In Parametri, fornire quanto segue:

Configurazione di rete per il VPC in cui viene eseguita l'applicazione consumer di flussi

- **`VPC`**: specifica il nome del VPC in cui verrà eseguita la funzione Lambda di polling.
- **`List of Subnet IDs`**: sottoreti in cui è stabilita un'interfaccia di rete. Aggiungere le sottoreti corrispondenti al cluster Neptune.
- **`List of Security Group Ids`**: specifica gli ID dei gruppi di sicurezza che concedono l'accesso in ingresso in scrittura al cluster database Neptune di origine.
- **`List of Route Table Ids`**: necessario per creare un endpoint Amazon DynamoDB nel VPC Neptune, se non è presente. È necessario fornire un elenco separato da virgole di ID tabella di routing associati alle sottoreti.
- **`Require to create Dynamo DB VPC Endpoint`**: valore booleano il cui valore predefinito è `true`. È necessario modificarlo e impostarlo su `false` solo se è già stato creato un endpoint DynamoDB nel VPC.
- **`Require to create Monitoring VPC Endpoint`**: valore booleano il cui valore predefinito è `true`. È necessario modificarlo in `false` solo se è già stato creato un endpoint di monitoraggio nel VPC.

Polling di flussi

- **`Application Name`**: in genere è possibile lasciare questo valore impostato sul valore predefinito (`NeptuneStream`). Se si utilizza un nome diverso, deve essere univoco.
- **`Memory size for Lambda Poller`**: consente di impostare le dimensioni della memoria disponibili per la funzione Lambda dello strumento di polling. Il valore predefinito è 2.048 megabyte.
- **`Lambda Runtime`**: linguaggio utilizzato nella funzione Lambda che recupera gli elementi dal flusso Neptune. Può essere impostato su `python3.9` o su `java8`.
- **`S3 Bucket having Lambda code artifacts`**: lascia questo parametro vuoto, a meno che non utilizzi una funzione di polling Lambda personalizzata che viene caricata da un bucket S3 diverso.
- **`S3 Key corresponding to Lambda Code artifacts`**: lascia questo parametro vuoto, a meno che non utilizzi una funzione di polling Lambda personalizzata.

- **`StartingCheckpoint`**: checkpoint iniziale per lo strumento per il polling dei flussi. L'impostazione predefinita è `0:0`, ovvero dall'inizio del flusso Neptune.
- **`StreamPollerInitialState`**: stato iniziale dello strumento per il polling. L'impostazione predefinita è `ENABLED`, quindi la replica del flusso verrà avviata al termine della creazione dell'intero stack.
- **`Logging level for Lambda`**: in generale, lascia il valore predefinito, `INFO`.
- **`Managed Policies for Lambda Execution`**: in generale, lascia vuoto a meno che non utilizzi una funzione di polling Lambda personalizzata.
- **`Stream Records Handler`**: in generale, lascia vuoto a meno che non utilizzi un gestore personalizzato per i record nei flussi Neptune.
- **`Maximum records Fetched from Stream`**: puoi utilizzare questo parametro per ottimizzare le prestazioni. Il valore predefinito (`100`) è un buon punto di partenza. Il massimo consentito è 10.000. Più alto è il numero, meno chiamate di rete sono necessarie per leggere i record dal flusso, ma maggiore è la memoria necessaria per elaborare i record.
- **`Max wait time between two Polls (in Seconds)`**: determina la frequenza con cui viene richiamato lo strumento di polling Lambda per il polling dei flussi Neptune. Impostare questo valore su 0 per il polling continuo. Il valore massimo è di 3.600 secondi (1 ora). Il valore predefinito (60 secondi) è un buon punto di partenza, a seconda della velocità con cui cambiano i dati del grafico.
- **`Maximum Continuous polling period (in Seconds)`**: consente di impostare un timeout per la funzione di polling Lambda. Dovrebbe essere compreso tra 5 e 900 secondi. Il valore predefinito (600 secondi) è un buon punto di partenza.
- **`Step Function Fallback Period`**— Il numero di step-function-fallback-period unità in attesa del poller, dopodiché la funzione step viene richiamata tramite Amazon CloudWatch Events per il ripristino in caso di errore. Il valore predefinito (5 minuti) è un buon punto di partenza.
- **`Step Function Fallback Period Unit`**: unità di tempo utilizzate per misurare il periodo `Step Function Fallback Period` precedente (minuti, ore, giorni). Il valore predefinito (minuti) è generalmente sufficiente.
- **`Data replication scope`**— Determina se replicare sia i nodi che i bordi o solo i nodi OpenSearch (questo vale solo per i dati del motore Gremlin). Il valore predefinito (Tutti) è generalmente un buon punto di partenza.
- **`Ignore OpenSearch missing document error`**— Contrassegno per determinare se un errore di documento mancante in OpenSearch può essere ignorato. Gli errori di documenti mancanti possono verificarsi raramente, ma richiedono un intervento manuale se non ignorati. Il valore predefinito (`True`) è generalmente un punto di partenza valido.

- **`Enable Non-String Indexing`**: flag per l'abilitazione o la disabilitazione dell'indicizzazione dei campi che non hanno un contenuto stringa. Se questo flag è impostato su`true`, i campi non di tipo stringa vengono indicizzati o OpenSearch, se`false`, vengono indicizzati solo i campi di tipo stringa. Il valore predefinito è `true`.
- **`Properties to exclude from being inserted into OpenSearch`**— Un elenco delimitato da virgole di proprietà o chiavi di predicati da escludere dall'indicizzazione. OpenSearch Se questo valore del parametro CFN viene lasciato vuoto, tutte le chiavi delle proprietà vengono indicizzate.
- **`Datatypes to exclude from being inserted into OpenSearch`**— Un elenco delimitato da virgole di tipi di dati di proprietà o predicati da escludere dall'indicizzazione. OpenSearch Se questo valore del parametro CFN viene lasciato vuoto, tutti i valori delle proprietà che possono essere convertiti in modo sicuro in tipi di dati vengono indicizzati. OpenSearch

Flusso Neptune

- **`Endpoint of source Neptune Stream`**: (obbligatorio). Può assumere uno dei due formati seguenti:
  - **`https://`*`your DB cluster`*`:`*`port`*`/propertygraph/stream`** (o il relativo alias, `https://`*`your DB cluster`*`:`*`port`*`/pg/stream`).
  - `https://`*`your DB cluster`*`:`*`port`*`/sparql/stream`
- **`Neptune Query Engine`**: scegli Gremlin o SPARQL.
- **`Is IAM Auth Enabled?`**: se il cluster database Neptune utilizza l'autenticazione IAM, imposta questo parametro su `true`.
- **`Neptune Cluster Resource Id`**: se il cluster database Neptune utilizza l'autenticazione IAM, imposta questo parametro sull'ID risorsa del cluster. L'ID risorsa non è lo stesso dell'ID cluster. Invece, assume il formato: `cluster-` seguito da 28 caratteri alfanumerici. Può essere trovato in Dettagli del cluster nella console Neptune.

Cluster di destinazione OpenSearch

- **`Endpoint for OpenSearch service`**— (Obbligatorio) Fornisci l'endpoint per il OpenSearch servizio nel tuo VPC.
- **`Number of Shards for OpenSearch Index`**: il valore predefinito (5) è generalmente un punto di partenza valido.

- **`Number of Replicas for OpenSearch Index`**: il valore predefinito (1) è generalmente un punto di partenza valido.
- **`Geo Location Fields for Mapping`**: se utilizzi campi di geolocalizzazione, elenca qui le chiavi delle proprietà.

Allarme

- **`Require to create Cloud watch Alarm`**— Impostalo su `true` se desideri creare un CloudWatch allarme per il nuovo stack.
- **`SNS Topic ARN for Cloudwatch Alarm Notifications`**— L'argomento SNS ARN in CloudWatch cui devono essere inviate le notifiche di allarme (necessario solo se gli allarmi sono abilitati).
- **`Email for Alarm Notifications`**: indirizzo e-mail a cui devono essere inviate le notifiche di allarme (necessario solo se gli allarmi sono abilitati).

Per la destinazione della notifica di allarme, è possibile aggiungere solo SNS, solo e-mail o sia SNS che e-mail.

Esegui il modello AWS CloudFormation

A questo punto è possibile completare il processo di provisioning di un'istanza dell'applicazione consumer di flussi Neptune come indicato di seguito:

1. Nella AWS CloudFormation pagina Specificare i dettagli dello stack, scegli Avanti.
2. Nella pagina Opzioni, scegli Avanti.
3. Nella pagina Revisione, seleziona la prima casella di controllo per accettare la creazione delle risorse IAM da parte di AWS CloudFormation . Seleziona la seconda casella di controllo per confermare `CAPABILITY_AUTO_EXPAND` per il nuovo stack.

> **Note**
>
> `CAPABILITY_AUTO_EXPAND` conferma in modo esplicito che, durante la creazione dello stack, le macro verranno ampliate senza revisione preventiva. Gli utenti spesso creano un set di modifiche da un modello elaborato, quindi le modifiche apportate dalle macro possono essere riesaminate prima dell'effettiva creazione dello stack. Per ulteriori

informazioni, consulta il funzionamento dell' AWS CloudFormation CreateStackAPI nell'AWS CloudFormation API Reference.

Quindi, scegli Crea.

Abilitazione della ricerca full-text sui database Neptune esistenti

Se è possibile sospendere i carichi di lavoro di scrittura

Quello illustrato di seguito è generalmente il modo migliore per attivare la ricerca full-text su un database Neptune esistente, a condizione che sia possibile sospendere i carichi di lavoro di scrittura. Richiede la creazione di un clone, l'abilitazione dei flussi utilizzando un parametro del cluster e il riavvio di tutte le istanze. La creazione di un clone è un'operazione relativamente veloce, quindi i tempi di inattività necessari sono limitati.

Procedura:

1. Arresta tutti i carichi di lavoro di scrittura sul database.
2. Abilita i flussi sul database (consulta Abilitazione dei flussi Neptune).
3. Crea un clone del database (consulta Clonazione del database in Neptune).
4. Riprendi i carichi di lavoro di scrittura.
5. Utilizza lo export-neptune-to-elasticsearchstrumento su github per eseguire una sincronizzazione unica dal database clonato al dominio. OpenSearch
6. Usa il modello AWS CloudFormation per la tua regione per avviare la sincronizzazione dal database originale con l'aggiornamento continuo (non richiede alcuna modifica alla configurazione del modello).
7. Elimina il database clonato e lo stack creato per lo strumento. AWS CloudFormation `export-neptune-to-elasticsearch`

Se non è possibile sospendere i carichi di lavoro di scrittura

Se non puoi permetterti di sospendere i carichi di lavoro di scrittura sul database, ecco un'alternativa che richiede tempi di inattività ancora inferiori rispetto all'approccio precedente, ma occorre prestare particolare attenzione:

1. Abilita i flussi sul database (consulta Abilitazione dei flussi Neptune).
2. Crea un clone del database (consulta Clonazione del database in Neptune).
3. Recupera gli ultimi `eventID` per i flussi nel database clonato eseguendo un comando di questo tipo sull'endpoint API Streams (consulta Chiamata alla REST API Flussi Neptune per ulteriori informazioni):

   ```
   curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
   iteratorType=LATEST"
   ```

   Prendi nota dei valori nei campi `commitNum` e `opNum` dell'oggetto `lastEventId` nella risposta.
4. Usa lo export-neptune-to-elasticsearchstrumento su github per eseguire una sincronizzazione unica dal database clonato al dominio. OpenSearch
5. Usa il modello AWS CloudFormation per la tua regione per avviare la sincronizzazione dal database originale con l'aggiornamento continuo.

   Apporta la seguente modifica durante la creazione dello stack: nella pagina dei dettagli dello stack, nella sezione Parametri imposta il valore del campo `StartingCheckpoint` su *`commitNum`*:*`opnum`* utilizzando i valori `commitNum` e `opNum` che annotati sopra.
6. Elimina il database clonato e lo stack creato per lo strumento. AWS CloudFormation `export-neptune-to-elasticsearch`

Aggiornamento dello strumento per il polling dei flussi

Per aggiornare lo strumento per il polling dei flussi con gli ultimi artefatti Lambda

Per aggiornare lo strumento per il polling dei flussi con gli ultimi artefatti Lambda, segui questa procedura:

1. In AWS Management Console, accedete allo stack principale principale AWS CloudFormation e selezionatelo. AWS CloudFormation
2. Seleziona l'opzione Aggiorna per lo stack.
3. Seleziona Sostituisci modello corrente.
4. Per l'origine del modello, scegli URL di Amazon S3 e immetti l'URL S3 seguente:

   ```
   https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
   neptune_to_elastic_search.json
   ```

5. Seleziona Avanti senza modificare alcun AWS CloudFormation parametro.
6. Scegliere Update Stack (Aggiorna stack).

Lo stack ora aggiornerà gli artefatti Lambda con quelli più recenti.

Estensione dello strumento per il polling dei flussi per il supporto di campi personalizzati

Lo strumento per il polling dei flussi corrente può essere facilmente esteso per scrivere codice personalizzato per la gestione di campi personalizzati, come illustrato in dettaglio in questo post del blog: Capture graph changes using Neptune Streams.

> **Note**
>
> Quando aggiungi un campo personalizzato OpenSearch, assicurati di aggiungere il nuovo campo come oggetto interno di un predicato (vediModello di dati Neptune per la ricerca full-text).

Disabilitazione e riabilitazione del processo di polling dei flussi

> **Warning**
>
> Presta attenzione durante la disabilitazione del processo di polling dei flussi. Può verificarsi una perdita di dati se il processo viene sospeso per un periodo più lungo della finestra di scadenza del flusso. La finestra predefinita è di 7 giorni, ma a partire dalla versione del motore 1.2.0.0, puoi impostare una finestra di scadenza del flusso personalizzata fino a un massimo di 90 giorni.

Disabilitazione (sospensione) del processo di polling dei flussi

1. Accedi AWS Management Console e apri la EventBridge console Amazon all'indirizzo https://console.aws.amazon.com/events/.
2. Nel riquadro di navigazione seleziona Regole.
3. Seleziona la regola il cui nome contiene il nome che hai fornito come nome dell'applicazione nel AWS CloudFormation modello che hai usato per configurare lo stream poller.

4. Scegliere Disabilita.
5. Apri la console Step Functions all'indirizzo https://console.aws.amazon.com/states/.
6. Seleziona la funzione della fase in esecuzione corrispondente al processo di polling dei flussi. Ancora una volta, il nome di quella funzione step contiene il nome che hai fornito come nome dell'applicazione nel AWS CloudFormation modello che hai usato per configurare lo stream poller. È possibile filtrare in base allo stato di esecuzione della funzione per visualizzare solo le funzioni In esecuzione.
7. Scegli Stop (Arresta).

Riabilitazione del processo di polling dei flussi

1. Accedi AWS Management Console e apri la EventBridge console Amazon all'indirizzo https://console.aws.amazon.com/events/.
2. Nel riquadro di navigazione seleziona Regole.
3. Seleziona la regola il cui nome contiene il nome che hai fornito come nome dell'applicazione nel AWS CloudFormation modello che hai usato per configurare lo stream poller.
4. Scegliere Disabilita. La regola dell'evento basata sull'intervallo pianificato specificato attiverà ora una nuova esecuzione della funzione di fase.

Replica in OpenSearch Serverless

A partire dal rilascio del motore 1.3.0.0, Amazon Neptune supporta l'utilizzo del servizio OpenSearch di Amazon Serverless per la ricerca di testo completo nelle query Gremlin e SPARQL.

Se stai eseguendo la replica in OpenSearch Serverless, aggiungi il ruolo di esecuzione dello strumento per il polling sul flusso Lambda alla policy di accesso ai dati per la raccolta OpenSearch Serverless. Il formato dell'ARN per il ruolo di esecuzione dello strumento per il polling sul flusso Lambda è il seguente:

```
arn:aws:iam::(account ID):role/stack-name-NeptuneOSReplication-
NeptuneStreamPollerExecu-(uuid)
```

Per ulteriori informazioni, consulta Controllo dell'accesso ai dati per Amazon OpenSearch Serverless.

Se hai abilitato il controllo granulare degli accessi nel cluster OpenSearch, devi anche abilitare l'autenticazione IAM nel database Neptune.

L'entità IAM (utente o ruolo) utilizzata per la connessione al database Neptune deve disporre delle autorizzazioni per Neptune e la raccolta OpenSearch Serverless. Questo significa che all'utente o al ruolo deve collegata una policy OpenSearch Serverless come questa:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::(account ID):root"
      },
      "Action": "aoss:APIAccessAll",
      "Resource": "arn:aws:aoss:(region):(account ID):collection/(collection ID)"
    }
  ]
}
```

Per ulteriori informazioni, consulta Dichiarazioni di policy di accesso ai dati IAM personalizzate per Amazon Neptune.

Esecuzione di query da un cluster OpenSearch con controllo granulare degli accessi attivato

Se hai abilitato il controllo granulare degli accessi nel cluster OpenSearch, devi abilitare l'autenticazione IAM anche nel database Neptune.

L'entità IAM (utente o ruolo) utilizzata per la connessione al database Neptune deve disporre delle autorizzazioni sia per Neptune che per il cluster OpenSearch. L'utente o il ruolo deve avere una policy del servizio OpenSearch come quella collegata:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-id:root"
      },
      "Action": "es:*",
```

```
      "Resource": "arn:aws:es:region:account-id:es-resource-id/*"
    }
  ]
}
```

Per ulteriori informazioni, consulta Dichiarazioni di policy di accesso ai dati IAM personalizzate per Amazon Neptune.

Utilizzo della sintassi di query Apache Lucene nelle query di ricerca full-text di Neptune

OpenSearch supporta l'utilizzo della sintassi Apache Lucene per le query query_string ed è particolarmente utile per passare più filtri in una query.

Neptune utilizza una struttura nidificata per archiviare le proprietà in un documento OpenSearch (consulta Modello di dati Neptune per la ricerca full-text). Quando si utilizza la sintassi Lucene, occorre usare i percorsi completi delle proprietà in questo modello nidificato.

Ecco un esempio per Gremlin:

```
g.withSideEffect("Neptune#fts.endpoint", "es_endpoint")
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V()
 .has("*", "Neptune#fts predicates.name.value:\"Jane Austin\" AND entity_type:Book")
```

Ecco un esempio per SPARQL:

```
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
  neptune-fts:config neptune-fts:endpoint 'http://localhost:9200 (http://
localhost:9200/)' .
  neptune-fts:config neptune-fts:queryType 'query_string' .
  neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*name.value:Ronak AND
 predicates.\\*foaf\\*surname.value:Sh*" .
  neptune-fts:config neptune-fts:field '*' .
  neptune-fts:config neptune-fts:return ?res .
}
```

Modello di dati Neptune per i dati di OpenSearch

Amazon Neptune utilizza una struttura di documenti JSON unificata per l'archiviazione dei dati SPARQL e Gremlin nel servizio OpenSearch. Ogni documento in OpenSearch corrisponde a un'entità e archivia tutte le informazioni pertinenti per l'entità stessa. Per Gremlin, i vertici (nodi) e gli archi sono considerati entità, quindi i documenti OpenSearch corrispondenti contengono informazioni su vertici, etichette e proprietà. Per SPARQL, i soggetti possono essere considerati entità, quindi i documenti OpenSearch corrispondenti contengono informazioni su tutte le coppie predicato-oggetto in un unico documento.

> **Note**
>
> L'implementazione della replica da Neptune a OpenSearch archivia solo i dati stringa. Tuttavia, è possibile modificarlo per memorizzare altri tipi di dati.

La struttura del documento JSON unificata è simile a questa:

```
{
  "entity_id": "Vertex Id/Edge Id/Subject URI",
  "entity_type": [List of Labels/rdf:type object value],
  "document_type": "vertex/edge/rdf-resource"
  "predicates": {
    "Property name or predicate URI": [
      {
        "value": "Property Value or Object Value",
        "graph": "(Only for Sparql) Named Graph Quad is present"
        "language": "(Only for Sparql) rdf:langString"
      },
      {
        "value": "Property Value 2/ Object Value 2",
      }
    ]
  }
}
```

- `entity_id`: ID univoco dell'entità che rappresenta il documento.
  - Per SPARQL, questo è l'URI del soggetto.
  - Per Gremlin, questo è `Vertex_ID` o `Edge_ID`.

- `entity_type`: rappresenta una o più etichette per un vertice o un arco o zero o più valori di predicato `rdf:type` per un soggetto.
- `document_type`: consente di specificare se il documento corrente rappresenta un vertice, un arco o una risorsa RDF.
- `predicates`: per Gremlin, archivia proprietà e valori per un vertice o un arco. Per SPARQL, memorizza coppie di oggetti-predicato.

  Il nome della proprietà assume il formato `properties.name.value` in OpenSearch. Per interrogarlo, devi nominarlo in quel modulo.
- `value` : valore della proprietà per Gremlin o valore di oggetto per SPARQL.
- `graph`: grafo denominato per SPARQL.
- `language`: tag del linguaggio per un valore letterale `rdf:langString` in SPARQL.

Documento OpenSearch di esempio per SPARQL

Dati

```
@prefix dt:   <http://example.org/datatype#> .
@prefix ex:   <http://example.org/> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

ex:simone   rdf:type     ex:Person                   ex:g1
ex:michael  rdf:type     ex:Person                   ex:g1
ex:simone   ex:likes     "spaghetti"                 ex:g1

ex:simone   ex:knows     ex:michael                  ex:g2   # Not stored in ES
ex:simone   ex:likes     "spaghetti"                 ex:g2
ex:simone   ex:status    "La vita è un sogno"@it     ex:g2

ex:simone   ex:age       "40"^^xsd:int               DG      # Not stored in ES
ex:simone   ex:dummy     "testData"^^dt:newDataType  DG
ex:simone   ex:hates     _:bnode                             # Not stored in ES
_:bnode     ex:means     "coding"                    DG      # Not stored in ES
```

Documenti

```
{
  "entity_id": "http://example.org/simone",
```

```
  "entity_type": ["http://example.org/Person"],
  "document_type": "rdf-resource"
  "predicates": {
    "http://example.org/likes": [
      {
        "value": "spaghetti",
        "graph": "http://example.org/g1"
      },
      {
        "value": "spaghetti",
        "graph": "http://example.org/g2"
      }
    ]
    "http://example.org/status": [
      {
        "value": "La vita è un sogno",
        "language": "it"       // Only present for rdf:langString
      }
    ]
  }
}
```

```
{
  "entity_id" : "http://example.org/michael",
  "entity_type" : ["http://example.org/Person"],
  "document_type": "rdf-resource"
}
```

Documento OpenSearch di esempio per Gremlin

Dati

```
# Vertex 1
simone   label    Person       <== Label
simone   likes    "spaghetti"  <== Property
simone   likes    "rice"       <== Property
simone   age      40           <== Property

# Vertex 2
michael  label    Person       <== Label

# Edge 1
simone  knows     michael      <== Edge
```

```
e1      updated  "2019-07-03"  <== Edge Property
e1      through  "company"     <== Edge Property
e1      since     10           <== Edge Property
```

Documenti

```
{
  "entity_id": "simone",
  "entity_type": ["Person"],
  "document_type": "vertex",
  "predicates": {
    "likes": [
      {
        "value": "spaghetti"
      },
      {
        "value": "rice"
      }
    ]
  }
}
```

```
{
  "entity_id" : "michael",
  "entity_type" : ["Person"],
  "document_type": "vertex"
}
```

```
{
  "entity_id": "e1",
  "entity_type": ["knows"],
  "document_type": "edge"
  "predicates": {
    "through": [
      {
        "value": "company"
      }
    ]
  }
}
```

Parametri per la ricerca full-text Neptune

Amazon Neptune utilizza i seguenti parametri per specificare le query full-text di OpenSearch in Gremlin e SPARQL:

- **`queryType`**: (obbligatorio). Tipo di query OpenSearch. Per un elenco dei tipi di query, consulta la documentazione di OpenSearch. Neptune supporta i seguenti tipi di query OpenSearch:
  - simple_query_string: restituisce documenti in base a una stringa di query specificata, utilizzando un parser con sintassi Lucene limitata ma con tolleranza agli errori. Questo è il tipo di query predefinito.

    Questa query utilizza una semplice sintassi per analizzare e dividere la stringa di query fornita in termini basati su operatori speciali. La query analizza quindi ogni termine in modo indipendente prima di restituire documenti corrispondenti.

    Mentre la sua sintassi è più limitata della query `query_string`, la query `simple_query_string` non restituisce errori per la sintassi non valida. Invece, ignora tutte le parti non valide della stringa di query.
  - match: la query `match` è la query standard per l'esecuzione di una ricerca full-text, incluse le opzioni per la corrispondenza fuzzy.
  - prefix : restituisce documenti che contengono un prefisso specifico in un campo specificato.
  - fuzzy: restituisce documenti che contengono termini simili al termine di ricerca, come misurato dalla distanza di edit di Levenshtein.

    Una distanza di edit è il numero di modifiche di un carattere necessarie per trasformare un termine in un altro. Queste modifiche possono includere:
    - Modifica di un carattere (ad esempio, da caso a casa).
    - Rimozione di un carattere (ad esempio da cassa a casa).
    - Inserimento di un carattere (ad esempio da cane a canne).
    - Trasposizione di due caratteri adiacenti (ad esempio da ramo ad armo).

    Per trovare termini simili, la query fuzzy crea un insieme di tutte le possibili varianti ed espansioni del termine di ricerca entro una distanza di edit specificata e quindi restituisce corrispondenze esatte per ciascuna di queste varianti.
  - term : restituisce documenti che contengono una corrispondenza esatta di un termine indicato in uno dei campi specificati.

È possibile utilizzare la query `term` per trovare documenti in base a un valore preciso, ad esempio un prezzo, un ID prodotto o un nome utente.

> ⚠ Warning
>
> Evitare di utilizzare la query di termine per i campi di testo. Per impostazione predefinita, OpenSearch modifica i valori dei campi di testo come parte dell'analisi, aumentando le difficoltà nel trovare corrispondenze esatte per i valori dei campi di testo.
> Per cercare i valori dei campi di testo, utilizzare invece la query di corrispondenza.

- query_string: restituisce documenti in base a una stringa di query specificata, utilizzando un parser con una sintassi rigorosa (sintassi Lucene).

  Questa query utilizza una sintassi per analizzare e dividere la stringa di query fornita in base agli operatori, ad esempio AND o NOT. La query analizza quindi ogni testo diviso in modo indipendente prima di restituire documenti corrispondenti.

  È possibile utilizzare la query `query_string` per creare una ricerca complessa che include caratteri jolly, ricerche in più campi e altro ancora. Sebbene versatile, la query è rigorosa e restituisce un errore se la stringa di query include una sintassi non valida.

  > ⚠ Warning
  >
  > Poiché restituisce un errore per qualsiasi sintassi non valida, non è consigliabile utilizzare la query `query_string` per le caselle di ricerca.
  > Se non è necessario supportare una sintassi di query, considerare l'utilizzo della query `match`. Se sono necessarie le funzionalità di una sintassi di query, utilizzare la query `simple_query_string`, che è meno rigorosa.

- **`field`**: campo di OpenSearch in base al cui eseguire la ricerca. Questo può essere omesso solo se `queryType` lo consente (come avviene per `simple_query_string` e `query_string`), nel qual caso la ricerca viene eseguita rispetto a tutti i campi. In Gremlin, è implicito.

  È possibile specificare più campi se la query lo consente, come avviene con `simple_query_string` e `query_string`.

- **`query`**: (obbligatorio). Query da eseguire su OpenSearch. Il contenuto di questo campo può variare in base al tipo di query. Diversi tipi di query accettano sintassi diverse come, ad esempio, nel caso di `Regexp`. In Gremlin, `query` è implicito.
- **`maxResults`**: numero massimo di risultati da restituire. Il valore predefinito è l'impostazione `index.max_result_window` di OpenSearch, che per impostazione predefinita è 10.000. Il parametro `maxResults` può specificare qualsiasi numero inferiore a quello.

  > **Important**
  >
  > Se si imposta `maxResults` su un valore superiore al valore `index.max_result_window` di OpenSearch e si tenta di recuperare più di `index.max_result_window` risultati, OpenSearch restituisce un errore `Result window is too large`. Tuttavia, Neptune gestisce questo scenario correttamente senza propagare l'errore. Tenere a mente ciò se stai cercando di recuperare più risultati `index.max_result_window`.

- **`minScore`**: punteggio minimo che un risultato di ricerca deve ottenere per essere restituito. Consulta la documentazione relativa alla pertinenza di OpenSearch per una spiegazione del punteggio dei risultati.
- **`batchSize`**: Neptune recupera sempre i dati in batch (le dimensioni predefinite del batch sono pari a 100). È possibile utilizzare questo parametro per ottimizzare le prestazioni. Le dimensioni del batch non possono superare l'impostazione `index.max_result_window` di OpenSearch, che per impostazione predefinita sono pari a 10.000.
- **`sortBy`**: parametro facoltativo che consente di ordinare i risultati restituiti da OpenSearch in base a una delle seguenti opzioni:
  - Un campo stringa specifico nel documento

    Ad esempio, in una query SPARQL, è possibile specificare:

    ```
    neptune-fts:config neptune-fts:sortBy foaf:name .
    ```

    In una query Gremlin simile, è possibile specificare:

    ```
    .withSideEffect('Neptune#fts.sortBy', 'name')
    ```

  - Un campo non stringa specifico (*`long`*, *`double`* e così via) nel documento

Tieni presente che quando si esegue l'ordinamento in base a un campo non stringa, è necessario aggiungere `.value` al nome del campo per differenziarlo da un campo stringa.

Ad esempio, in una query SPARQL, è possibile specificare:

```
    neptune-fts:config neptune-fts:sortBy foaf:name.value .
```

In una query Gremlin simile, è possibile specificare:

```
    .withSideEffect('Neptune#fts.sortBy', 'name.value')
```

- `score`: ordina in base al punteggio di corrispondenza (valore predefinito).

  Se il parametro `sortOrder` è presente ma `sortBy` non è presente, i risultati vengono ordinati `score` in base all'ordine specificato da `sortOrder`.
- `id`: ordina in base all'ID, ovvero l'URI del soggetto in SPARQL o l'ID vertice o arco in Gremlin.

  Ad esempio, in una query SPARQL, è possibile specificare:

  ```
      neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_id' .
  ```

  In una query Gremlin simile, è possibile specificare:

  ```
      .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_id')
  ```
- `label`: ordina in base all'etichetta.

  Ad esempio, in una query SPARQL, è possibile specificare:

  ```
      neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_type' .
  ```

  In una query Gremlin simile, è possibile specificare:

  ```
      .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_type')
  ```
- `doc_type`: ordina in base al tipo di documento (ovvero SPARQL o Gremlin).

  Ad esempio, in una query SPARQL, è possibile specificare:

```
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.document_type' .
```

In una query Gremlin simile, è possibile specificare:

```
    .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.document_type')
```

Per impostazione predefinita, i risultati di OpenSearch non sono ordinati e l'ordine non è deterministico, di conseguenza la stessa query può restituire gli elementi in un ordine diverso ogni volta che viene eseguita. Per questo motivo, se il set di risultati è maggiore di `max_result_window`, un sottoinsieme piuttosto differente del totale dei risultati potrebbe essere restituito ogni volta che viene eseguita una query. Ordinando, tuttavia, è possibile rendere i risultati di diverse esecuzioni comparabili in modo più diretto.

Se nessun parametro `sortOrder` accompagna `sortBy`, viene utilizzato l'ordine discendente (`DESC`) dal più grande al più piccolo.

- **`sortOrder`**: parametro facoltativo che consente di specificare se i risultati di OpenSearch sono ordinati dal valore minimo al valore massimo o dal valore massimo al valore minimo (impostazione predefinita):

  - `ASC`: ordine crescente, dal valore minimo al valore massimo.
  - `DESC`: ordine decrescente, dal valore massimo al valore minimo.

    Questo è il valore predefinito, utilizzato quando il parametro `sortBy` è presente ma non è specificato alcun `sortOrder`.

  Se né `sortBy` né `sortOrder` sono specificati, i risultati di OpenSearch non sono ordinati per impostazione predefinita.

Indicizzazione OpenSearch non tipo stringa in Amazon Neptune.

L'indicizzazione OpenSearch non stringa in Amazon Neptune consente di replicare valori non stringa per i predicati in OpenSearch tramite lo strumento per il polling dei flussi. Tutti i valori dei predicati che possono essere convertiti in modo sicuro in una mappatura o un tipo di dati OpenSearch corrispondente vengono quindi replicati in OpenSearch.

Per abilitare l'indicizzazione non stringa su un nuovo stack, il flag `Enable Non-String Indexing` nel modello AWS CloudFormation deve essere impostato su `true`. Questa è l'impostazione predefinita. Per aggiornare uno stack esistente per supportare l'indicizzazione non stringa, consulta Aggiornamento di uno stack esistente sotto.

> **Note**
>
> - È preferibile non abilitare l'indicizzazione non stringa su versioni del motore precedenti alla versione **`1.0.4.2`**.
> - Le query di OpenSearch che utilizzano espressioni regolari per i nomi di campo che corrispondono a più campi, alcuni con valori stringa e altri con valori non stringa, restituiscono un errore. Lo stesso accade se le query di ricerca full-text in Neptune sono di quel tipo.
> - Quando si esegue l'ordinamento in base a un campo non stringa, è necessario aggiungere ".value" al nome del campo per differenziarlo da un campo stringa.

Indice

Aggiornamento di uno stack di ricerca full-text di Neptune esistente per supportare l'indicizzazione non stringa

Se stai già utilizzando la ricerca full-text di Neptune, ecco la procedura da seguire per supportare l'indicizzazione non stringa:

1. Arresta la funzione Lambda dello strumento per il polling dei flussi. Questo garantisce che nessun nuovo aggiornamento venga copiato durante l'esportazione. A tale scopo, disabilita la regola degli eventi cloud che richiama la funzione Lambda:

   - Nella AWS Management Console passa a CloudWatch.
   - Seleziona Regole.
   - Scegli la regola con il nome dello strumento per il polling dei flussi Lambda.
   - Seleziona disabilita per disabilitare temporaneamente la regola.

2. Elimina l'indice Neptune corrente in OpenSearch. Usa la query `curl` seguente per eliminare l'indice `amazon_neptune` dal cluster OpenSearch:

   ```
   curl -X DELETE "your OpenSearch endpoint/amazon_neptune"
   ```

3. Avvia un'esportazione una tantum da Neptune a OpenSearch. A questo punto è consigliabile configurare un nuovo stack OpenSearch, per selezionare nuovi artefatti per lo strumento di polling che esegue l'esportazione.

   Segui i passaggi elencati qui in GitHub per avviare l'esportazione una tantum dei dati di Neptune in OpenSearch.

4. Aggiorna gli artefatti Lambda per lo strumento per il polling dei flussi esistente. Una volta completata correttamente l'esportazione dei dati di Neptune in OpenSearch, segui questa procedura:

   - In AWS Management Console, andare su AWS CloudFormation.
   - Scegli lo stack AWS CloudFormation padre principale.
   - Seleziona l'opzione Aggiorna per lo stack.
   - Seleziona Sostituisci il modello corrente dalle opzioni.
   - Per il modello di origine, seleziona URL Amazon S3.
   - Per l'URL Amazon S3, immetti:

```
https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
neptune_to_elastic_search.json
```

- Seleziona Avanti senza modificare alcun parametro di AWS CloudFormation.
- Seleziona Aggiorna stack. AWS CloudFormation sostituirà gli artefatti del codice Lambda per il polling dei flussi con gli artefatti più recenti.

5. Avvia nuovamente lo strumento per il polling dei flussi. A tale scopo, abilita la regola di CloudWatch appropriata:

   - Nella AWS Management Console passa a CloudWatch.
   - Seleziona Regole.
   - Scegli la regola con il nome dello strumento per il polling dei flussi Lambda.
   - Seleziona Abilita.

Filtro dei campi indicizzati nella ricerca full-text di Neptune

Nei dettagli del modello AWS CloudFormation sono presenti due campi che consentono di specificare chiavi di proprietà o predicato o tipi di dati da escludere dall'indicizzazione OpenSearch:

Filtra per nome della proprietà o del predicato

È possibile utilizzare il parametro facoltativo AWS CloudFormation del modello denominato `Properties to exclude from being inserted into Elastic Search Index` per fornire un elenco separato da virgole di chiavi di proprietà o predicato da escludere dall'indicizzazione OpenSearch.

Ad esempio, supponi di impostare questo parametro su `bob`:

```
"Properties to exclude from being inserted into Elastic Search Index" : bob
```

In tal caso, il record del flusso della seguente query di aggiornamento Gremlin verrebbe eliminato anziché essere incluso nell'indice:

```
g.V("1").property("bob", "test")
```

Allo stesso modo, potresti impostare il parametro su `http://my/example#bob`:

```
"Properties to exclude from being inserted into Elastic Search Index" : http://my/
example#bob
```

In tal caso, il record del flusso della seguente query di aggiornamento SPARQL verrebbe eliminato anziché essere incluso nell'indice:

```
PREFIX ex: <http://my/example#>
INSERT DATA { ex:s1 ex:bob "test"}.
```

Se non inserisci nulla in questo parametro del modello AWS CloudFormation, verranno indicizzate tutte le chiavi di proprietà non altrimenti escluse.

Filtra per tipo di valore di proprietà o predicato

È possibile utilizzare il parametro facoltativo AWS CloudFormation del modello denominato `Datatypes to exclude from being inserted into Elastic Search Index` per fornire un elenco separato da virgole di tipi di dati della proprietà o del predicato da escludere dall'indicizzazione OpenSearch.

Per SPARQL, non è necessario elencare l'URI completo del tipo XSD, è sufficiente specificare il token del tipo di dati. I token del tipo di dati validi che puoi elencare sono:

- `string`
- `boolean`
- `float`
- `double`
- `dateTime`
- `date`
- `time`
- `byte`
- `short`
- `int`
- `long`
- `decimal`
- `integer`
- `nonNegativeInteger`

- `nonPositiveInteger`
- `negativeInteger`
- `unsignedByte`
- `unsignedShort`
- `unsignedInt`
- `unsignedLong`

Per Gremlin, i tipi di dati validi da elencare sono:

- `string`
- `date`
- `bool`
- `byte`
- `short`
- `int`
- `long`
- `float`
- `double`

Ad esempio, supponi di impostare questo parametro su `string`:

```
"Datatypes to exclude from being inserted into Elastic Search Index" : string
```

In tal caso, il record del flusso della seguente query di aggiornamento Gremlin verrebbe eliminato anziché essere incluso nell'indice:

```
g.V("1").property("myStringval", "testvalue")
```

Allo stesso modo, potresti impostare il parametro su `int`:

```
"Datatypes to exclude from being inserted into Elastic Search Index" : int
```

In tal caso, il record del flusso della seguente query di aggiornamento SPARQL verrebbe eliminato anziché essere incluso nell'indice:

```
PREFIX ex: <http://my/example#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
INSERT DATA { ex:s1 ex:bob "11"^^xsd:int }.
```

Se non specifichi nulla in questo parametro del modello AWS CloudFormation, verranno indicizzate tutte le proprietà i cui valori possono essere convertiti in modo sicuro in equivalenti OpenSearch. I tipi elencati non sono supportati dal linguaggio di query vengono ignorati.

Mappatura dei tipi di dati SPARQL e Gremlin a OpenSearch

Le nuove mappature dei tipi di dati in OpenSearch vengono create in base al tipo di dati utilizzato nella proprietà o nell'oggetto. Poiché alcuni campi contengono valori di tipi diversi, la mappatura iniziale può escludere alcuni valori del campo.

La mappatura dei tipi di dati di Neptune ai tipi di dati di OpenSearch è la seguente:

Tipi SPARQL	Tipi Gremlin	Tipi OpenSearch
`XSD:int`	`byte`	`long`
`XSD:unsignedInt`	`short`	
`XSD:integer`	`int`	
`XSD:byte`	`long`	
`XSD:unsignedByte`		
`XSD:short`		
`XSD:unsignedShort`		
`XSD:long`		
`XSD:unsignedLong`		
`XSD:float`	`float`	`double`
`XSD:double`	`double`	
`XSD:decimal`		

Tipi SPARQL	Tipi Gremlin	Tipi OpenSearch
`XSD:boolean`	`bool`	`boolean`
`XSD:datetime` `XSD:date`	`date`	`date`
`XSD:string` `XSD:time`	`string`	`text`
Tipo di dati personalizzato	N/D	`text`
Qualsiasi altro tipo di dati	N/D	`text`

Ad esempio, la seguente query di aggiornamento Gremlin causa l'aggiunta di una nuova mappatura per "newField" a OpenSearch, ovvero `{ "type" : "double" }`:

```
g.V("1").property("newField" 10.5)
```

Allo stesso modo, la seguente query di aggiornamento SPARQL causa l'aggiunta di una nuova mappatura per "ex:byte" a OpenSearch, ovvero `{ "type" : "long" }`:

```
PREFIX ex: <http://my/example#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>

INSERT DATA { ex:test ex:byte "123"^^xsd:byte }.
```

> **Note**
>
> Come si può notare, un elemento mappato da Neptune a OpenSearch, in OpenSearch potrebbe essere convertito in un tipo di dati diverso da quello presente in Neptune. Tuttavia, in OpenSearch esiste un campo di testo esplicito, "datatype", che registra il tipo di dati dell'elemento in Neptune.

Convalida delle mappature dei dati

I dati vengono replicati in OpenSearch da Neptune mediante questo processo:

- Se una mappatura per il campo in questione è già presente in OpenSearch:
  - Se i dati possono essere convertiti in modo sicuro nella mappatura esistente utilizzando le regole di convalida dei dati, archivia il campo in OpenSearch.
  - In caso contrario, elimina il record di aggiornamento del flusso corrispondente.
- Se non è presente una mappatura esistente per il campo in questione, trova un tipo di dati OpenSearch corrispondente al tipo di dati del campo in Neptune.
  - Se i dati del campo possono essere convertiti in modo sicuro nel tipo di dati OpenSearch utilizzando le regole di convalida dei dati, archivia la nuova mappatura e i dati del campo in OpenSearch.
  - In caso contrario, elimina il record di aggiornamento del flusso corrispondente.

I valori vengono convalidati rispetto a tipi OpenSearch equivalenti o mappature OpenSearch esistenti anziché ai tipi Neptune. Ad esempio, la convalida del valore `"123"` in `"123"^^xsd:int` viene eseguita rispetto al tipo `long` anziché al tipo `int`.

Sebbene Neptune tenti di replicare tutti i dati in OpenSearch, esistono casi in cui i tipi di dati in OpenSearch sono totalmente diversi da quelli di Neptune, e in questi scenari i record vengono ignorati anziché essere indicizzati in OpenSearch.

Ad esempio, in Neptune una proprietà può avere più valori di tipi diversi, mentre in OpenSearch un campo deve avere lo stesso tipo in tutto l'indice.

Abilitando i log di debug, è possibile visualizzare i record eliminati durante l'esportazione da Neptune a OpenSearch. Un esempio di voce del log di debug è:

```
Dropping Record : Data type not a valid Gremlin type
<Record>
```

I tipi di dati vengono convalidati come segue:

- **`text`**: tutti i valori di Neptune possono essere mappati in modo sicuro al testo in OpenSearch.
- **`long`**: le regole seguenti per i tipi di dati di Neptune si applicano quando il tipo di mappatura OpenSearch è long (negli esempi seguenti, si suppone che il tipo di mappatura di `"testLong"` sia `long`):

- `boolean`: il tipo non è valido, non può essere convertito e il record di aggiornamento del flusso corrispondente viene eliminato.

  Esempi di tipi Gremlin non validi sono:

  ```
  "testLong" : true.
  "testLong" : false.
  ```

  Esempi di tipi SPARQL non validi sono:

  ```
  ":testLong" : "true"^^xsd:boolean
  ":testLong" : "false"^^xsd:boolean
  ```

- `datetime`: il tipo non è valido, non può essere convertito e il record di aggiornamento del flusso corrispondente viene eliminato.

  Un esempio di tipo Gremlin non valido è:

  ```
  ":testLong" :  datetime('2018-11-04T00:00:00').
  ```

  Un esempio di tipo SPARQL non valido è:

  ```
  ":testLong" : "2016-01-01"^^xsd:date
  ```

- `float`, `double`, o `decimal`: se il valore in Neptune è un numero intero a 64 bit, è valido ed è archiviato in OpenSearch come valore long, ma se ha una parte frazionaria, o è un valore `NaN` o `INF`, o è maggiore di 9.223.372.036.854.775.807 o minore di -9.223.372.036.854.775.808, non è valido e il record di aggiornamento del flusso corrispondente viene eliminato.

  Esempi di tipi Gremlin validi sono:

  ```
  "testLong" :  145.0.
  ":testLong" :  123
  ":testLong" :  -9223372036854775807
  ```

  Esempi di tipi SPARQL validi sono:

  ```
  ":testLong" : "145.0"^^xsd:float
  ":testLong" :  145.0
  ```

```
":testLong" : "145.0"^^xsd:double
":testLong" : "145.0"^^xsd:decimal
":testLong" : "-9223372036854775807"
```

Esempi di tipi Gremlin non validi sono:

```
"testLong" :  123.45
":testLong" :  9223372036854775900
```

Esempi di tipi SPARQL non validi sono:

```
":testLong" :  123.45
":testLong" :  9223372036854775900
":testLong" : "123.45"^^xsd:float
":testLong" : "123.45"^^xsd:double
":testLong" : "123.45"^^xsd:decimal
```

- `string`: se il valore in Neptune è una rappresentazione in formato stringa di un numero intero a 64 bit, è valido e viene convertito in un valore `long` in OpenSearch. Qualsiasi altro valore stringa non è valido per una mappatura `long` in Elasticseearch e il record di aggiornamento del flusso corrispondente viene eliminato.

  Esempi di tipi Gremlin validi sono:

  ```
  "testLong" :  "123".
  ":testLong" :  "145.0"
  ":testLong" :  "-9223372036854775807"
  ```

  Esempi di tipi SPARQL validi sono:

  ```
  ":testLong" : "145.0"^^xsd:string
  ":testLong" : "-9223372036854775807"^^xsd:string
  ```

  Esempi di tipi Gremlin non validi sono:

  ```
  "testLong" :  "123.45"
  ":testLong" :  "9223372036854775900"
  ":testLong" :  "abc"
  ```

Esempi di tipi SPARQL non validi sono:

```
  ":testLong" : "123.45"^^xsd:string
  ":testLong" : "abc"
  ":testLong" : "9223372036854775900"^^xsd:string
```

- **`double`**: se il tipo di mappatura OpenSearch è `double`, si applicano le seguenti regole (in questo caso, si presuppone che il campo "testDouble" abbia una mappatura `double` in OpenSearch):
  - `boolean`: il tipo non è valido, non può essere convertito e il record di aggiornamento del flusso corrispondente viene eliminato.

    Esempi di tipi Gremlin non validi sono:

    ```
      "testDouble" : true.
      "testDouble" : false.
    ```

    Esempi di tipi SPARQL non validi sono:

    ```
      ":testDouble" : "true"^^xsd:boolean
      ":testDouble" : "false"^^xsd:boolean
    ```

  - `datetime`: il tipo non è valido, non può essere convertito e il record di aggiornamento del flusso corrispondente viene eliminato.

    Un esempio di tipo Gremlin non valido è:

    ```
      ":testDouble" :  datetime('2018-11-04T00:00:00').
    ```

    Un esempio di tipo SPARQL non valido è:

    ```
      ":testDouble" : "2016-01-01"^^xsd:date
    ```

  - Valore `NaN` o `INF` a virgola mobile: se il valore in SPARQL è un valore `NaN` o `INF` a virgola mobile, non è valido e il record di aggiornamento del flusso corrispondente viene eliminato.

    Esempi di tipi SPARQL non validi sono:

    ```
    "  :testDouble" : "NaN"^^xsd:float
      ":testDouble" : "NaN"^^double
    ```

```
":testDouble" : "INF"^^double
":testDouble" : "-INF"^^double
```

- numero o stringa numerica: se il valore in Neptune è qualsiasi altro numero o rappresentazione stringa numerica di un numero che può essere espresso come un valore `double`, è valido e viene convertito in un valore `double` in OpenSearch. Qualsiasi altro valore stringa non è valido per una mappatura `double` in OpenSearch e il record di aggiornamento del flusso corrispondente viene eliminato.

  Esempi di tipi Gremlin validi sono:

  ```
  "testDouble" :  123
  ":testDouble" :  "123"
  ":testDouble" :  145.67
  ":testDouble" :  "145.67"
  ```

  Esempi di tipi SPARQL validi sono:

  ```
  ":testDouble" :  123.45
  ":testDouble" :  145.0
  ":testDouble" : "123.45"^^xsd:float
  ":testDouble" : "123.45"^^xsd:double
  ":testDouble" : "123.45"^^xsd:decimal
  ":testDouble" : "123.45"^^xsd:string
  ```

  Un esempio di tipo Gremlin non valido è:

  ```
  ":testDouble" :  "abc"
  ```

  Esempio SPARQL non validi sono:

  ```
  ":testDouble" : "abc"
  ```

- **`date`**: se il tipo di mappatura OpenSearch è `date`, i valori `date` e `dateTime` in Neptune sono validi, così come qualsiasi valore stringa che può essere analizzato correttamente in un formato `dateTime`.

  Esempi validi in Gremlin o SPARQL sono:

  ```
  Date(2016-01-01)
  ```

```
"2016-01-01" "
2003-09-25T10:49:41"
"2003-09-25T10:49"
"2003-09-25T10"
"20030925T104941-0300"
"20030925T104941"
"2003-Sep-25" "
Sep-25-2003"
"2003.Sep.25"
"2003/09/25"
"2003 Sep 25" "
Wed, July 10, '96"
"Tuesday, April 12, 1952 AD 3:30:42pm PST"
"123"
"-123"
"0"
"-0"
"123.00"
"-123.00"
```

Esempi non validi sono:

```
123.45
True
"abc"
```

Query OpenSearch non stringa di esempio in Neptune

Neptune attualmente non supporta direttamente le query di intervallo di OpenSearch. Tuttavia, è possibile ottenere lo stesso effetto utilizzando la sintassi Lucene e query-type="query_string", come si può osservare nelle seguenti query di esempio.

1. Recupera tutti i vertici con età maggiore di 30 anni e nome che inizia con "Si"

In Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.age.value:>30 && predicates.name.value:Si*');
```

In SPARQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*age.value:>30 AND
 predicates.\\*foaf\\*name.value:Si*" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

In questo caso, per brevità viene usato "`\\*foaf\\*age` anziché l'URI completo. Questa espressione regolare recupererà tutti i campi che contengono `foaf` e `age` nell'URI.

2. Recupera tutti i nodi con età compresa tra 10 e 50 anni e un nome con una corrispondenza fuzzy con "Ronka"

In Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.age.value:[10 TO 50] AND
 predicates.name.value:Ronka~');
```

In SPARQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*age.value:[10 TO 50] AND
 predicates.\\*foaf\\*name.value:Ronka~" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

3. Recupera tutti i nodi con un timestamp che rientra negli ultimi 25 giorni

In Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:>now-25d');
```

In SPARQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\
\*timestamp.value:>now-25d~" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

4. Recupera tutti i nodi con un timestamp che rientra in un anno e mese specifico

In Gremlin, usando espressioni date math nella sintassi Lucene, per dicembre 2020:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:>2020-12');
```

Alternativa a Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:[2020-12 TO 2021-01]');
```

Esecuzione di query di ricerca full-text in Amazon Neptune

In una query che include la ricerca full-text, Neptune tenta di eseguire le chiamate di ricerca full-text prima di altre parti della query. Questo riduce il numero di chiamate a OpenSearch e nella maggior parte dei casi migliora significativamente le prestazioni. Tuttavia, questa non è affatto una regola rigida. Esistono situazioni, ad esempio, in cui `PatternNode` o `UnionNode` può precedere una chiamata di ricerca full-text.

Si consideri la seguente query Gremlin su un database contenente 100.000 istanze di `Person`:

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .hasLabel('Person')
 .has('name', 'Neptune#fts marcello~');
```

Se questa query fosse eseguita nell'ordine in cui vengono visualizzati i passaggi, verrebbero trasmesse 100.000 soluzioni a OpenSearch, causando centinaia di chiamate OpenSearch. In effetti, Neptune chiama prima OpenSearch e poi unisce i risultati con i risultati di Neptune. Nella maggior parte dei casi, questo è molto più veloce rispetto all'esecuzione della query nell'ordine originale.

È possibile impedire questo riordinamento dell'esecuzione del passaggio di query utilizzando l'hint di query noReordering:

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .withSideEffect('Neptune#noReordering', true)
 .hasLabel('Person')
 .has('name', 'Neptune#fts marcello~');
```

In questo secondo caso, il passaggio `.hasLabel` viene eseguito per primo e il passaggio `.has('name', 'Neptune#fts marcello~')` per secondo.

Per un altro esempio, si consideri una query SPARQL sullo stesso tipo di dati:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT ?person WHERE {
  ?person rdf:type foaf:Person .
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mike' .
    neptune-fts:config neptune-fts:return ?person .
```

```
  }
}
```

Anche in questo caso, Neptune esegue prima la parte `SERVICE` della query, poi unisce i risultati con i dati relativi a `Person`. È possibile eliminare questo comportamento usando l'hint di query joinOrder:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?person WHERE {
  hint:Query hint:joinOrder "Ordered" .
  ?person rdf:type foaf:Person .
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mike' .
    neptune-fts:config neptune-fts:return ?person .
  }
}
```

Ancora una volta, nella seconda query le parti vengono eseguite nell'ordine in cui appaiono nella query.

Query SPARQL di esempio che utilizzano la ricerca full-text in Neptune

Di seguito sono riportate alcune query SPARQL di esempio che utilizzano la ricerca full-text in Amazon Neptune.

Esempio di query match SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'match' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'michael' .
    neptune-fts:config neptune-fts:return ?res .
  }
```

```
}
```

Esempio di query prefix SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'prefix' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mich' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Esempio di query fuzzy SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'fuzzy' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mikael' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Esempio di query term SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'term' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'Dr. Kunal' .
    neptune-fts:config neptune-fts:return ?res .
```

```
  }
}
```

Esempio di query query_string SPARQL

Questa query specifica più campi.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ OR rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:field foaf:surname .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Esempio di query simple_query_string SPARQL

La query seguente specifica i campi utilizzando il carattere jolly ('*').

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'simple_query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Esempio di query di ordinamento per campo stringa SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
```

```
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy foaf:name .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Esempio di query di ordinamento per campo non stringa SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name.value .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy dc:date.value .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Esempio di query di ordinamento per ID SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_id' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Esempio di query di ordinamento per etichetta SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_type' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Esempio di query di ordinamento per tipo di documento SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.document_type' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Esempio di utilizzo della sintassi Lucene in SPARQL

La sintassi Lucene è supportata solo per le query `query_string` in OpenSearch.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
```

```
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'predicates.\\foaf\\name.value:micheal AND
 predicates.\\foaf\\surname.value:sh' .
    neptune-fts:config neptune-fts:field '' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Utilizzo della ricerca full-text di Neptune nelle query Gremlin

`NeptuneSearchStep` abilita le query di ricerca full-text per la parte di un attraversamento Gremlin che non viene convertito in fasi di Neptune. Ad esempio, considerare una query come la seguente:

```
g.withSideEffect("Neptune#fts.endpoint", "your-es-endpoint-URL")
  .V()
      .tail(100)
      .has("name", "Neptune#fts mark*")           <== # Limit the search on name
```

Questa query viene convertita nel seguente attraversamento ottimizzato in Neptune:

```
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=4}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [NeptuneTailGlobalStep(100),
 NeptuneTinkerpopTraverserConverterStep, NeptuneSearchStep {
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=your-OpenSearch-endpoint-URL}
    }
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=your-OpenSearch-endpoint-URL}
    }
}]
```

I seguenti esempi sono query Gremlin sui dati delle rotte aeree:

Query **match** di base senza distinzione tra maiuscole e minuscole Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'match')
  .V().has("city","Neptune#fts dallas")

==>v[186]
==>v[8]
```

Query **match** Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'match')
  .V().has("city","Neptune#fts southampton")
     .local(values('code','city').fold())
     .limit(5)

==>[SOU, Southampton]
```

Query **fuzzy** Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has("city","Neptune#fts allas~").values('city').limit(5)

==>Dallas
==>Dallas
==>Walla Walla
==>Velas
==>Altai
```

Query fuzzy **query_string** Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has("city","Neptune#fts allas~").values('city').limit(5)
```

```
==>Dallas
==>Dallas
```

Query di espressioni regolari **query_string** Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has("city","Neptune#fts /[dp]allas/").values('city').limit(5)

==>Dallas
==>Dallas
```

Query ibrida Gremlin

Questa query utilizza un indice interno Neptune e l'indice OpenSearch nella stessa query.

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has("region","GB-ENG")
      .has('city','Neptune#fts L*')
      .values('city')
      .dedup()
      .limit(10)

==>London
==>Leeds
==>Liverpool
==>Land's End
```

Esempio di ricerca full-text semplice Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has('desc','Neptune#fts regional municipal')
      .local(values('code','desc').fold())
      .limit(100)

==>[HYA, Barnstable Municipal Boardman Polando Field]
==>[SPS, Sheppard Air Force Base-Wichita Falls Municipal Airport]
```

```
==>[ABR, Aberdeen Regional Airport]
==>[SLK, Adirondack Regional Airport]
==>[BFD, Bradford Regional Airport]
==>[EAR, Kearney Regional Airport]
==>[ROT, Rotorua Regional Airport]
==>[YHD, Dryden Regional Airport]
==>[TEX, Telluride Regional Airport]
==>[WOL, Illawarra Regional Airport]
==>[TUP, Tupelo Regional Airport]
==>[COU, Columbia Regional Airport]
==>[MHK, Manhattan Regional Airport]
==>[BJI, Bemidji Regional Airport]
==>[HAS, Hail Regional Airport]
==>[ALO, Waterloo Regional Airport]
==>[SHV, Shreveport Regional Airport]
==>[ABI, Abilene Regional Airport]
==>[GIZ, Jizan Regional Airport]
==>[USA, Concord Regional Airport]
==>[JMS, Jamestown Regional Airport]
==>[COS, City of Colorado Springs Municipal Airport]
==>[PKB, Mid Ohio Valley Regional Airport]
```

Query Gremlin che utilizza **`query_string`** con gli operatori "+" e "-"

Sebbene il tipo di query `query_string` sia molto meno permissiva rispetto al tipo `simple_query_string` predefinito, consente query più precise. La prima query seguente utilizza `query_string`, mentre la seconda utilizza il valore predefinito `simple_query_string`:

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
 . V().has('desc','Neptune#fts +London -(Stansted|Gatwick)')
      .local(values('code','desc').fold())
      .limit(10)

==>[LHR, London Heathrow]
==>[YXU, London Airport]
==>[LTN, London Luton Airport]
==>[SEN, London Southend Airport]
==>[LCY, London City Airport]
```

Si noti come `simple_query_string` negli esempi seguenti ignori gli operatori "+" e "-"

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has('desc','Neptune#fts +London -(Stansted|Gatwick)')
      .local(values('code','desc').fold())
      .limit(10)

==>[LHR, London Heathrow]
==>[YXU, London Airport]
==>[LGW, London Gatwick]
==>[STN, London Stansted Airport]
==>[LTN, London Luton Airport]
==>[SEN, London Southend Airport]
==>[LCY, London City Airport]
==>[SKG, Thessaloniki Macedonia International Airport]
==>[ADB, Adnan Menderes International Airport]
==>[BTV, Burlington International Airport]
```

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has('desc','Neptune#fts +(regional|municipal) -(international|bradford)')
      .local(values('code','desc').fold())
      .limit(10)

==>[CZH, Corozal Municipal Airport]
==>[MMU, Morristown Municipal Airport]
==>[YBR, Brandon Municipal Airport]
==>[RDD, Redding Municipal Airport]
==>[VIS, Visalia Municipal Airport]
==>[AIA, Alliance Municipal Airport]
==>[CDR, Chadron Municipal Airport]
==>[CVN, Clovis Municipal Airport]
==>[SDY, Sidney Richland Municipal Airport]
==>[SGU, St George Municipal Airport]
```

Query **`query_string`** Gremlin con gli operatori **AND** e **OR**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has('desc','Neptune#fts (St AND George) OR (St AND Augustin)')
      .local(values('code','desc').fold())
```

```
      .limit(10)

==>[YIF, St Augustin Airport]
==>[STG, St George Airport]
==>[SGO, St George Airport]
==>[SGU, St George Municipal Airport]
```

Query **`term`** Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'term')
  .V().has("SKU","Neptune#fts ABC123DEF9")
      .local(values('code','city').fold())
      .limit(5)

==>[AUS, Austin]
```

Query **`prefix`** Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'prefix')
  .V().has("icao","Neptune#fts ka")
      .local(values('code','icao','city').fold())
      .limit(5)

==>[AZO, KAZO, Kalamazoo]
==>[APN, KAPN, Alpena]
==>[ACK, KACK, Nantucket]
==>[ALO, KALO, Waterloo]
==>[ABI, KABI, Abilene]
```

Utilizzo della sintassi Lucene in Neptune Gremlin

In Neptune Gremlin è possibile scrivere query molto potenti utilizzando la sintassi di query Lucene. Tieni presente che la sintassi Lucene è supportata solo per le query `query_string` in OpenSearch.

Prendere come esempio i seguenti dati:

```
g.addV("person")
```

```
        .property(T.id, "p1")
        .property("name", "simone")
        .property("surname", "rondelli")

g.addV("person")
        .property(T.id, "p2")
        .property("name", "simone")
        .property("surname", "sengupta")

g.addV("developer")
        .property(T.id, "p3")
        .property("name", "simone")
        .property("surname", "rondelli")
```

Utilizzando la sintassi di Lucene, che viene richiamata quando `queryType` è `query_string`, è possibile cercare questi dati per nome e cognome come segue:

```
g.withSideEffect("Neptune#fts.endpoint", "es_endpoint")
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts predicates.name.value:simone AND
 predicates.surname.value:rondelli")

==> v[p1], v[p3]
```

Si noti che nel passaggio `has()` precedente, il campo viene sostituito da `"*"`. In realtà, qualsiasi valore inserito in quel campo viene sovrascritto dai campi a cui si accede all'interno della query. Si accede al campo nome utilizzando `predicates.name.value,` perché questo è il modo in cui il modello di dati è strutturato.

È possibile cercare per nome, cognome ed etichetta, come segue:

```
g.withSideEffect("Neptune#fts.endpoint", getEsEndpoint())
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts predicates.name.value:simone AND
 predicates.surname.value:rondelli AND entity_type:person")

==> v[p1]
```

All'etichetta si accede utilizzando `entity_type`, ancora una volta perché è così che il modello di dati è strutturato.

È inoltre possibile includere le condizioni di nidificazione:

```
g.withSideEffect("Neptune#fts.endpoint", getEsEndpoint())
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts (predicates.name.value:simone AND
 predicates.surname.value:rondelli AND entity_type:person) OR
 predicates.surname.value:sengupta")

==> v[p1], v[p2]
```

Inserimento di un grafo TinkerPop moderno

```
g.addV('person').property(T.id, '1').property('name', 'marko').property('age', 29)
 .addV('personr').property(T.id, '2').property('name', 'vadas').property('age', 27)
 .addV('software').property(T.id, '3').property('name', 'lop').property('lang', 'java')
 .addV('person').property(T.id, '4').property('name', 'josh').property('age', 32)
 .addV('software').property(T.id, '5').property('name', 'ripple').property('lang',
 'java')
 .addV('person').property(T.id, '6').property('name', 'peter').property('age', 35)

g.V('1').as('a').V('2').as('b').addE('knows').from('a').to('b').property('weight',
 0.5f).property(T.id, '7')
 .V('1').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.4f).property(T.id, '9')
 .V('4').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.4f).property(T.id, '11')
 .V('4').as('a').V('5').as('b').addE('created').from('a').to('b').property('weight',
 1.0f).property(T.id, '10')
 .V('6').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.2f).property(T.id, '12')
 .V('1').as('a').V('4').as('b').addE('knows').from('a').to('b').property('weight',
 1.0f).property(T.id, '8')
```

Esempio di valore per ordinamento per campo stringa

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'name')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Esempio di valore per ordinamento per campo non stringa

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'age.value')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Esempio di valore per ordinamento per campo ID

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
.withSideEffect('Neptune#fts.queryType', 'query_string')
.withSideEffect('Neptune#fts.sortOrder', 'asc')
.withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_id')
.V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Esempio di valore per ordinamento per campo etichetta

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_type')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Esempio di valore per ordinamento per campo **`document_type`**

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.document_type')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Risoluzione dei problemi di ricerca full-text di Neptune

> **Note**
>
> Se hai abilitato il controllo granulare degli accessi nel cluster OpenSearch, devi abilitare l'autenticazione IAM anche nel database Neptune.

Per diagnosticare i problemi con la replica da Neptune a OpenSearch, consulta File di log Amazon CloudWatch per la funzione Lambda dello strumento di polling. Questi log forniscono informazioni dettagliate sul numero di record letti dal flusso e il numero di record replicati correttamente in OpenSearch.

È inoltre possibile cambiare il livello di registrazione di log per la funzione Lambda modificando la variabile di ambiente `LoggingLevel`.

> **Note**
>
> Se la variabile `LoggingLevel` è impostata su `DEBUG`, puoi visualizzare dettagli aggiuntivi, come i record dei flussi eliminati e il motivo dell'eliminazione, mentre StreamPoller replica i dati da Neptune a OpenSearch. Questa impostazione può essere utile in caso di record mancanti.

Anche l'applicazione consumer dei flussi di Neptune pubblica due metriche su CloudWatch per assisterti nella diagnosi dei problemi:

- `StreamRecordsProcessed`: il numero di record elaborati dall'applicazione per unità di tempo. Utile nel tracciare la velocità di esecuzione dell'applicazione.
- `StreamLagTime`: la differenza di tempo in millisecondi tra l'ora corrente e l'ora del commit di un record del flusso in fase di elaborazione. Questo parametro indica il ritardo dell'applicazione consumer.

Inoltre, tutte le metriche relative al processo di replica sono esposte in un pannello di controllo in CloudWatch con lo stesso nome di quello fornito da `ApplicationName` quando è stata creata un'istanza dell'applicazione utilizzando il modello CloudWatch.

È anche possibile scegliere di creare un allarme CloudWatch che viene attivato ogni volta che il polling non riesce più di due volte di seguito. Per fare ciò, impostare il campo `CreateCloudWatchAlarm` su `true` quando si crea un'istanza dell'applicazione. Quindi specificare gli indirizzi e-mail che si desidera ricevano una notifica quando viene attivato l'allarme.

Risoluzione dei problemi di un processo che non riesce durante la lettura dei record dal flusso

Se un processo non riesce durante la lettura dei record dal flusso, verificare di disporre di quanto segue:

- Il flusso è abilitato nel cluster.
- L'endpoint del flusso Neptune è nel formato corretto:
  - Per Gremlin o openCypher: `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/propertygraph/stream` o il relativo alias, `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/pg/stream`
  - Per SPARQL: `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/sparql/stream`
- L'endpoint DynamoDB è configurato per il VPC.
- L'endpoint di monitoraggio è configurato per le sottoreti del VPC.

Risoluzione dei problemi di un processo che ha esito negativo durante la scrittura dei dati in OpenSearch

Se un processo ha esito negativo durante la scrittura di record in OpenSearch, verifica quanto segue:

- È installata la versione 7.1 o successiva di Elasticsearch o la versione 2.3 o successiva di OpenSearch.
- OpenSearch è accessibile alla funzione Lambda dello strumento di polling nel VPC.
- La policy di sicurezza collegata a OpenSearch consente le richieste HTTP/HTTPS in ingresso.

Risoluzione dei problemi di mancata sincronizzazione tra Neptune e OpenSearch in una configurazione di replica esistente

Usa la procedura seguente per ripristinare la sincronizzazione tra un database Neptune e un dominio OpenSearch con i dati più recenti in caso di problemi di mancata sincronizzazione derivanti da un danneggiamento dei dati o un'eccezione `ExpiredStreamException`.

Tieni presente che questo approccio elimina tutti i dati nel dominio OpenSearch e li sincronizza nuovamente dallo stato corrente del database Neptune, quindi non è necessario ricaricare i dati nel database Neptune.

1. Disabilita il processo di replica come descritto in Disabilitazione (sospensione) del processo di polling dei flussi.
2. Elimina l'indice Neptune nel dominio OpenSearch utilizzando il seguente comando:

   ```
   curl -X DELETE "(your OpenSearch endpoint)/amazon_neptune"
   ```

3. Crea un clone del database (consulta Clonazione del database in Neptune).
4. Recupera gli ultimi `eventID` per i flussi nel database clonato eseguendo un comando di questo tipo sull'endpoint API Streams (consulta Chiamata alla REST API Flussi Neptune per ulteriori informazioni):

   ```
   curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
   iteratorType=LATEST"
   ```

   Prendi nota dei valori nei campi `commitNum` e `opNum` dell'oggetto `lastEventId` nella risposta.
5. Usa lo strumento export-neptune-to-elasticsearch in GitHub per eseguire una sincronizzazione una tantum dal database clonato al dominio OpenSearch.
6. Vai alla tabella DynamoDB per lo stack di replica. Il nome della tabella corrisponderà al Nome applicazione specificato nel modello AWS CloudFormation (l'impostazione predefinita è `NeptuneStream`) con un suffisso `-LeaseTable`. In altre parole, il nome predefinito della tabella è `NeptuneStream-LeaseTable`.

   È possibile esplorare le righe della tabella mediante la scansione, in quanto nella tabella deve essere presente una sola riga. Apporta le seguenti modifiche utilizzando i valori `opNum` e `commitNum` annotati in precedenza:

   - Modifica il valore del campo `checkpoint` della tabella con il valore annotato per `commitNum`.
   - Modifica il valore del campo `checkpointSubSequenceNumber` della tabella con il valore annotato per `opNum`.
7. Riabilita il processo di replica come descritto in Riabilitazione del processo di polling dei flussi.
8. Elimina il database clonato e lo stack AWS CloudFormation creato per lo strumento `export-neptune-to-elasticsearch`.

Utilizzo delle funzioni AWS Lambda in Amazon Neptune

Le funzioni AWS Lambda hanno molti utilizzi nelle applicazioni Amazon Neptune. Qui vengono fornite indicazioni generali su come utilizzare le funzioni Lambda con i driver e le varianti del linguaggio Gremlin più diffusi e per esempi specifici di funzioni Lambda scritte in Java, JavaScript e Python.

> **Note**
>
> Il modo migliore per utilizzare le funzioni Lambda con Neptune è cambiato con i rilasci recenti del motore. Neptune lasciava aperte le connessioni inattive per periodi prolungati dopo il riciclo di un contesto di esecuzione Lambda, causando potenzialmente una perdita di risorse sul server. Per mitigare questo problema, era consigliabile aprire e chiudere una connessione con ogni invocazione Lambda. A partire dalla versione del motore 1.0.3.0, tuttavia, il timeout per le connessioni inattive è stato ridotto per evitare perdite delle connessioni dopo il riciclo di un contesto di esecuzione Lambda inattivo, quindi ora è consigliabile utilizzare una singola connessione per la durata del contesto di esecuzione. Ciò deve includere la gestione degli errori e il codice boilerplate per backoff e nuovo tentativo per gestire le connessioni che vengono chiuse in modo imprevisto.

Gestione delle connessioni WebSocket Gremlin nelle funzioni AWS Lambda

Se si utilizza una variante del linguaggio Gremlin per l'esecuzione di query su Neptune, il driver si connette al database utilizzando una connessione WebSocket. I WebSocket sono progettati per supportare scenari di connessione client-server di lunga durata. AWS Lambda, al contrario, è progettato per supportare esecuzioni di durata relativamente breve e senza stato. Questa discrepanza nella filosofia di progettazione può causare alcuni problemi imprevisti durante l'utilizzo di Lambda per l'esecuzione di query su Neptune.

Una funzione AWS Lambda viene eseguita in un contesto di esecuzione che la isola dalle altre funzioni. Il contesto di esecuzione viene creato la prima volta che la funzione viene richiamata e può essere riutilizzato per le successive invocazioni della stessa funzione.

Tuttavia, un contesto di esecuzione non viene mai utilizzato per gestire più invocazioni simultanee della funzione. Se la funzione viene richiamata simultaneamente da più client, Lambda crea un

contesto di esecuzione aggiuntivo per ogni istanza della funzione. Tutti questi nuovi contesti di esecuzione possono a loro volta essere riutilizzati per le successive invocazioni della funzione.

A un certo punto, Lambda ricicla i contesti di esecuzione, in particolare se sono rimasti inattivi per qualche tempo. AWS Lambda espone il ciclo di vita del contesto di esecuzione, incluse le fasi `Init`, `Invoke` e `Shutdown`, tramite estensioni Lambda. Usando queste estensioni, è possibile scrivere codice che esegue la pulizia delle risorse esterne, come le connessioni al database, quando il contesto di esecuzione viene riciclato.

Una best practice comune prevede l'apertura della connessione al database all'esterno della funzione del gestore Lambda affinché possa essere riutilizzata con ogni chiamata al gestore. Se a un certo punto la connessione al database si interrompe, puoi riconnetterti dall'interno del gestore. Tuttavia, questo approccio comporta il rischio di perdite di connessione. Se una connessione inattiva rimane aperta a lungo dopo la distruzione di un contesto di esecuzione, gli scenari di invocazione Lambda intermittenti possono causare la perdita graduale delle connessioni ed esaurire le risorse del database.

I limiti di connessione e i timeout di connessione di Neptune hanno subito modifiche con i nuovi rilasci del motore. In precedenza, ogni istanza supportava fino a 60.000 connessioni WebSocket. Ora, il numero massimo di connessioni WebSocket simultanee per istanza Neptune varia a seconda del tipo di istanza.

Inoltre, a partire dal rilascio del motore 1.0.3.0, Neptune ha ridotto il timeout di inattività per le connessioni da un'ora a circa 20 minuti. Se un client non chiude una connessione, questa viene chiusa automaticamente dopo un timeout di inattività di 20-25 minuti. AWS Lambda non documenta le durate del contesto di esecuzione, ma gli esperimenti mostrano che il nuovo timeout di connessione Neptune è allineato con i timeout dei contesti di esecuzione Lambda inattivi. Quando un contesto di esecuzione inattivo viene riciclato, esistono sono buone probabilità che la sua connessione sia già stata chiusa da Neptune o che venga chiusa subito dopo.

Raccomandazioni per l'utilizzo di AWS Lambda con Gremlin in Amazon Neptune

È consigliabile utilizzare un'unica connessione e origine di attraversamento del grafo per l'intera durata di un contesto di esecuzione Lambda, anziché una per ogni invocazione di funzione (ogni invocazione di funzione gestisce solo una richiesta client). Poiché le richieste client simultanee vengono gestite da diverse istanze di funzione in esecuzione in contesti di esecuzione separati,

non è necessario mantenere un pool di connessioni per gestire le richieste simultanee all'interno di un'istanza di funzione. Se il driver Gremlin in uso ha un pool di connessioni, configuralo affinché utilizzi una sola connessione.

Per gestire gli errori di connessione, utilizza la logica di ripetizione dei tentativi per ogni query. Anche se l'obiettivo è mantenere una singola connessione per tutta la durata di un contesto di esecuzione, eventi di rete imprevisti possono causare l'interruzione improvvisa della connessione. Tali errori di connessione si manifestano come errori diversi a seconda del driver in uso. È necessario codificare la funzione Lambda per gestire questi problemi di connessione e tentare una riconnessione, se necessario.

Alcuni driver Gremlin gestiscono automaticamente le riconnessioni. Il driver Java, ad esempio, tenta automaticamente di ristabilire la connettività a Neptune per conto del codice client. Con questo driver, è sufficiente che il codice della funzione esegua il backoff e ripeta la query. I driver JavaScript e Python, al contrario, non implementano alcuna logica di riconnessione automatica, quindi con questi driver il codice della funzione deve provare a riconnettersi dopo il backoff e ripetere la query solo dopo aver ristabilito la connessione.

Gli esempi di codice qui riportati includono la logica di riconnessione anziché presupporre che questa venga gestita dal client.

Raccomandazioni per l'utilizzo delle richieste di scrittura Gremlin in Lambda

Se la funzione Lambda modifica i dati del grafo, valuta l'adozione di una strategia di backoff e nuovo tentativo per gestire le seguenti eccezioni:

- **`ConcurrentModificationException`**: la semantica delle transazioni di Neptune prevede che le richieste di scrittura abbiano talvolta esito negativo con un'eccezione `ConcurrentModificationException`. In queste situazioni, prova un meccanismo di ripetizione dei tentativi esponenziale basato sul backoff.
- **`ReadOnlyViolationException`**: poiché la topologia del cluster può cambiare in qualsiasi momento a seguito di eventi pianificati o non pianificati, le responsabilità di scrittura potrebbero essere trasferite da un'istanza del cluster a un'altra. Se il codice della funzione tenta di inviare una richiesta di scrittura a un'istanza che non è più l'istanza primaria (scrittura), la richiesta ha esito negativo e restituisce un'eccezione `ReadOnlyViolationException`. In questo caso, chiudi la connessione esistente, riconnettiti all'endpoint del cluster e ripeti la richiesta.

Inoltre, se utilizzi una strategia di backoff e nuovo tentativo per gestire i problemi relativi alle richieste di scrittura, valuta l'implementazione di query idempotenti per le richieste di creazione e aggiornamento (ad esempio, utilizzando fold().coalesce().unfold().

Raccomandazioni per l'utilizzo delle richieste di lettura Gremlin in Lambda

Se nel cluster sono presenti una o più repliche di lettura, è consigliabile bilanciare le richieste di lettura tra queste repliche. Un'opzione prevede l'utilizzo dell'endpoint di lettura. L'endpoint di lettura bilancia le connessioni tra le repliche anche se la topologia del cluster cambia a seguito dell'aggiunta o della rimozione di repliche o della promozione di una replica affinché diventi la nuova istanza primaria.

L'utilizzo dell'endpoint di lettura può tuttavia comportare un utilizzo non uniforme delle risorse del cluster in alcune circostanze. L'endpoint di lettura funziona cambiando periodicamente l'host verso cui punta la voce DNS. Se un client apre molte connessioni prima che la voce DNS cambi, tutte le richieste di connessione vengono inviate a una singola istanza Neptune. Potrebbe essere il caso con uno scenario Lambda a velocità di trasmissione effettiva elevata in cui un gran numero di richieste simultanee alla funzione Lambda causa la creazione di più contesti di esecuzione, ognuno con la propria connessione. Se queste connessioni vengono create quasi simultaneamente, è probabile che puntino tutte alla stessa replica nel cluster e che continuino a puntare a quella replica fino al riciclo dei contesti di esecuzione.

Un modo per distribuire le richieste tra le istanze consiste nel configurare la funzione Lambda per connettersi a un endpoint dell'istanza, scelto casualmente da un elenco di endpoint di istanze di replica, anziché all'endpoint di lettura. Lo svantaggio di questo approccio è che richiede che il codice Lambda gestisca le modifiche nella topologia del cluster tramite il monitoraggio del cluster e l'aggiornamento dell'elenco degli endpoint ogni volta che cambia l'appartenenza al cluster.

Se stai scrivendo una funzione Lambda Java che deve bilanciare le richieste di lettura tra le istanze del cluster, puoi utilizzare il client Gremlin per Amazon Neptune, un client Gremlin Java che riconosce la topologia del cluster e che distribuisce equamente connessioni e richieste su un set di istanze in un cluster Neptune. Questo post del blog include una funzione Lambda Java di esempio che utilizza il client Gremlin per Amazon Neptune.

Fattori che possono rallentare l'avvio a freddo delle funzioni Lambda Gremlin in Neptune

La prima volta che una funzione AWS Lambda viene richiamata viene definita avvio a freddo. Esistono diversi fattori che possono aumentare la latenza di un avvio a freddo:

- Assicurati di assegnare memoria sufficiente alla funzione Lambda. La compilazione durante un avvio a freddo per una funzione Lambda può essere significativamente più lenta rispetto a quanto avviene in EC2, in quanto AWS Lambda alloca i cicli della CPU in modo lineare in proporzione alla memoria assegnata alla funzione. Con 1.792 MB di memoria, una funzione riceve l'equivalente di una vCPU completa (un secondo vCPU di crediti al secondo). L'impatto dell'assegnazione di memoria insufficiente per ricevere cicli di CPU adeguati è particolarmente importante per le funzioni Lambda di grandi dimensioni scritte in Java.
- Tieni presente che l'abilitazione dell'autenticazione del database IAM può rallentare un avvio a freddo: L'autenticazione del database AWS Identity and Access Management (IAM) può anche rallentare gli avvii a freddo, in particolare se la funzione Lambda deve generare una nuova chiave di firma. Questa latenza influisce solo sull'avvio a freddo e non sulle richieste successive, perché una volta che l'autenticazione del database IAM ha stabilito le credenziali di connessione, Neptune verifica solo periodicamente che siano ancora valide.

Utilizzo di AWS CloudFormation per creare una funzione Lambda da usare in Neptune

Puoi utilizzare un modello AWS CloudFormation per creare una funzione AWS Lambda che può accedere a Neptune.

1. Per avviare lo stack della funzione Lambda nella console AWS CloudFormation, scegli uno dei pulsanti Avvia lo stack nella seguente tabella.

Region	Vista	Visualizzazione in Designer	Avvia
Stati Uniti orientali (Virginia settentrionale)	Visualizzazione	Visualizzazione in Designer	
Stati Uniti orientali (Ohio)	Visualizzazione	Visualizzazione in Designer	

Region	Vista	Visualizzazione in Designer	Avvia
Stati Uniti occidentali (California settentrionale)	Visualizzazione	Visualizzazione in Designer	
Stati Uniti occidentali (Oregon)	Visualizzazione	Visualizzazione in Designer	
Canada (Centrale)	Visualizzazione	Visualizzazione in Designer	
Sud America (San Paolo)	Visualizzazione	Visualizzazione in Designer	
Europa (Stoccolma)	Visualizzazione	Visualizzazione in Designer	
Europa (Irlanda)	Visualizzazione	Visualizzazione in Designer	
Europa (Londra)	Visualizzazione	Visualizzazione in Designer	
Europa (Parigi)	Visualizzazione	Visualizzazione in Designer	
Europa (Francoforte)	Visualizzazione	Visualizzazione in Designer	
Medio Oriente (Bahrein)	Visualizzazione	Visualizzazione in Designer	
Medio Oriente (Emirati Arabi Uniti)	Visualizzazione	Visualizzazione in Designer	
Israele (Tel Aviv)	Visualizzazione	Visualizzazione in Designer	

Region	Vista	Visualizzazione in Designer	Avvia
Africa (Città del Capo)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Hong Kong)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Tokyo)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Seul)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Singapore)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Sydney)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Mumbai)	Visualizzazione	Visualizzazione in Designer	
Cina (Pechino)	Visualizzazione	Visualizzazione in Designer	
Cina (Ningxia)	Visualizzazione	Visualizzazione in Designer	
AWS GovCloud (Stati Uniti occidentali)	Visualizzazione	Visualizzazione in Designer	
AWS GovCloud (Stati Uniti orientali)	Visualizzazione	Visualizzazione in Designer	

2. Nella pagina Select Template (Seleziona modello), selezionare Next (Avanti).
3. Imposta le opzioni seguenti nella pagina Specify details (Specifica dettagli):

a. Scegli il runtime Lambda in base al linguaggio che vuoi usare nella funzione Lambda. Questi modelli di AWS CloudFormation attualmente supportano le seguenti lingue:

   - Python 3.9 (mappato a `python39` nell'URL Amazon S3)
   - NodeJS 18 (mappato a `nodejs18x` nell'URL Amazon S3)
   - Ruby 2.5 (mappato a `ruby25` nell'URL Amazon S3)

b. Specifica l'endpoint del cluster Neptune e il numero di porta appropriati.

c. Specifica il gruppo di sicurezza Neptune appropriato.

d. Specifica i parametri di sottorete Neptune appropriati.

4. Seleziona Successivo.
5. Nella pagina Opzioni, scegli Avanti.
6. Nella pagina Revisione, seleziona la prima casella di controllo per accettare la creazione delle risorse IAM da parte di AWS CloudFormation.

   Quindi, scegli Crea.

Se si ha necessità di apportare modifiche personalizzate al runtime Lambda, è possibile scaricarne uno generico da una posizione Amazon S3 nella propria regione:

```
https://s3.Amazon region.amazonaws.com/aws-neptune-customer-samples-Amazon region/
lambda/runtime-language/lambda_function.zip.
```

Per esempio:

```
https://s3.us-west-2.amazonaws.com/aws-neptune-customer-samples-us-west-2/lambda/
python36/lambda_function.zip
```

Esempi di funzioni AWS Lambda per Amazon Neptune

Le seguenti funzioni AWS Lambda di esempio, scritte in Java, JavaScript e Python, illustrano l'upsert di un singolo vertice con un ID generato casualmente utilizzando l'idioma `fold().coalesce().unfold()`.

Gran parte del codice contenuto in ogni funzione è codice boilerplate, responsabile della gestione delle connessioni e dei tentativi di connessione e query in caso di errore. La vera logica

dell'applicazione e la query Gremlin sono implementate rispettivamente nei metodi `doQuery()` e `query()`. Se usi questi esempi come base per le funzioni Lambda, puoi concentrarti sulla modifica di `doQuery()` e `query()`.

Le funzioni sono configurate per ripetere le query non riuscite 5 volte, con l'attesa di 1 secondo tra un tentativo e l'altro.

Le funzioni richiedono che i valori siano presenti nelle seguenti variabili di ambiente Lambda:

- **`NEPTUNE_ENDPOINT`**: endpoint del cluster database Neptune. Per Python, sarà `neptuneEndpoint`
- **`NEPTUNE_PORT`**: porta di Neptune. Per Python, sarà `neptunePort`
- **`USE_IAM`** (`true` o `false`): se nel database è abilitata l'autenticazione AWS Identity and Access Management (IAM), imposta la variabile di ambiente `USE_IAM` su `true`. In questo modo, la funzione Lambda firmerà le richieste di connessione a Neptune tramite SigV4. Per tali richieste di autenticazione IAM del database, assicurati che al ruolo di esecuzione della funzione Lambda sia collegata una policy IAM appropriata che consenta alla funzione di connettersi al cluster database Neptune (consulta Tipi di policy IAM).

Esempio di funzione Lambda Java per Amazon Neptune

Di seguito sono illustrati alcuni aspetti da considerare sulle funzioni AWS Lambda Java:

- Il driver Java mantiene il proprio pool di connessioni, che non è necessario, quindi configura l'oggetto `Cluster` con `minConnectionPoolSize(1)` e `maxConnectionPoolSize(1)`.
- La creazione dell'oggetto `Cluster` può essere lenta in quanto vengono creati uno o più serializzatori (per impostazione predefinita Gyro, più un altro se è stato configurato per formati di output aggiuntivi come `binary`). La creazione dell'istanza di questi può richiedere tempi prolungati.
- Il pool di connessioni viene inizializzato con la prima richiesta. A questo punto, il driver configura lo stack `Netty`, alloca i buffer di byte e crea una chiave di firma se utilizzi l'autenticazione del database IAM. Tutto ciò può aumentare la latenza di avvio a freddo.
- Il pool di connessioni del driver Java monitora la disponibilità degli host del server e tenta automaticamente di riconnettersi in caso di errore della connessione. Avvia un'attività in background per tentare di ristabilire la connessione. Usa `reconnectInterval( )` per configurare l'intervallo tra i tentativi di riconnessione. Mentre il driver tenta di riconnettersi, la funzione Lambda può semplicemente provare a ripetere la query.

Se l'intervallo tra i tentativi è inferiore all'intervallo tra i tentativi di riconnessione, i nuovi tentativi su una connessione non riuscita avranno nuovamente esito negativo perché l'host è considerato non disponibile. Non si applica ai nuovi tentativi per un'eccezione `ConcurrentModificationException`.

- Usa Java 8 anziché Java 11. Le ottimizzazioni `Netty` in Java 11 non sono abilitate per impostazione predefinita.
- Questo esempio utilizza Retry4j per i tentativi.
- Per utilizzare il driver di firma `Sigv4` nella funzione Lambda Java, consulta i requisiti per le dipendenze in Connessione a Neptune tramite Java e Gremlin con firma di Signature Version 4.

> **Warning**
>
> L'executor `CallExecutor` di Retry4j potrebbe non essere thread-safe. Prevedi che ogni thread utilizzi un'istanza `CallExecutor` dedicata.

```
package com.amazonaws.examples.social;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestStreamHandler;
import com.evanlennick.retry4j.CallExecutor;
import com.evanlennick.retry4j.CallExecutorBuilder;
import com.evanlennick.retry4j.Status;
import com.evanlennick.retry4j.config.RetryConfig;
import com.evanlennick.retry4j.config.RetryConfigBuilder;
import org.apache.tinkerpop.gremlin.driver.Cluster;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer;
import org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteConnection;
import org.apache.tinkerpop.gremlin.driver.ser.Serializers;
import org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource;
import org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversalSource;
import org.apache.tinkerpop.gremlin.structure.T;

import java.io.*;
import java.time.temporal.ChronoUnit;
import java.util.HashMap;
import java.util.Map;
```

```
import java.util.Random;
import java.util.concurrent.Callable;
import java.util.function.Function;

import static java.nio.charset.StandardCharsets.UTF_8;
import static org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.__.addV;
import static org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.__.unfold;

public class MyHandler implements RequestStreamHandler {

  private final GraphTraversalSource g;
  private final CallExecutor<Object> executor;
  private final Random idGenerator = new Random();

  public MyHandler() {

    this.g = AnonymousTraversalSource
            .traversal()
            .withRemote(DriverRemoteConnection.using(createCluster()));


    this.executor = new CallExecutorBuilder<Object>()
            .config(createRetryConfig())
            .build();

  }

  @Override
  public void handleRequest(InputStream input,
                            OutputStream output,
                            Context context) throws IOException {

    doQuery(input, output);
  }

  private void doQuery(InputStream input, OutputStream output) throws IOException {
    try {

      Map<String, Object> args = new HashMap<>();
      args.put("id", idGenerator.nextInt());

      String result = query(args);

      try (Writer writer = new BufferedWriter(new OutputStreamWriter(output, UTF_8))) {
```

```
          writer.write(result);
      }

    } finally {
        input.close();
        output.close();
    }
  }

  private String query(Map<String, Object> args) {
    int id = (int) args.get("id");

    @SuppressWarnings("unchecked")
    Callable<Object> query = () -> g.V(id)
      .fold()
      .coalesce(
        unfold(),
        addV("Person").property(T.id, id))
      .id().next();

    Status<Object> status = executor.execute(query);

    return status.getResult().toString();
  }

  private Cluster createCluster() {
    Cluster.Builder builder = Cluster.build()

.addContactPoint(System.getenv("NEPTUNE_ENDPOINT"))

.port(Integer.parseInt(System.getenv("NEPTUNE_PORT")))
                                    .enableSsl(true)
                                    .minConnectionPoolSize(1)
                                    .maxConnectionPoolSize(1)
                                    .serializer(Serializers.GRAPHBINARY_V1D0)
                                    .reconnectInterval(2000);

  if (Boolean.parseBoolean(getOptionalEnv("USE_IAM", "true"))) {
    // For versions of TinkerPop 3.4.11 or higher:
    builder.handshakeInterceptor( r ->
      {
        NeptuneNettyHttpSigV4Signer sigV4Signer = new
NeptuneNettyHttpSigV4Signer(region, new DefaultAWSCredentialsProviderChain());
        sigV4Signer.signRequest(r);
```

```
        return r;
      }
    )

    // Versions of TinkerPop prior to 3.4.11 should use the following approach.
    // Be sure to adjust the  imports to include:
    //   import org.apache.tinkerpop.gremlin.driver.SigV4WebSocketChannelizer;
    //   builder = builder.channelizer(SigV4WebSocketChannelizer.class);

    return builder.create();
  }

  private RetryConfig createRetryConfig() {
    return new RetryConfigBuilder().retryOnCustomExceptionLogic(retryLogic())
                                   .withDelayBetweenTries(1000, ChronoUnit.MILLIS)
                                   .withMaxNumberOfTries(5)
                                   .withFixedBackoff()
                                   .build();
  }

  private Function<Exception, Boolean> retryLogic() {
    return e -> {
      StringWriter stringWriter = new StringWriter();
      e.printStackTrace(new PrintWriter(stringWriter));
      String message = stringWriter.toString();

      // Check for connection issues
      if ( message.contains("Timed out while waiting for an available host") ||
           message.contains("Timed-out waiting for connection on Host") ||
           message.contains("Connection to server is no longer active") ||
           message.contains("Connection reset by peer") ||
           message.contains("SSLEngine closed already") ||
           message.contains("Pool is shutdown") ||
           message.contains("ExtendedClosedChannelException") ||
           message.contains("Broken pipe")) {
        return true;
      }

      // Concurrent writes can sometimes trigger a ConcurrentModificationException.
      // In these circumstances you may want to backoff and retry.
      if (message.contains("ConcurrentModificationException")) {
          return true;
      }
```

```
      // If the primary fails over to a new instance, existing connections to the old
 primary will
      // throw a ReadOnlyViolationException. You may want to back and retry.
      if (message.contains("ReadOnlyViolationException")) {
          return true;
      }

      return false;
    };
  }

  private String getOptionalEnv(String name, String defaultValue) {
    String value = System.getenv(name);
    if (value != null && value.length() > 0) {
      return value;
    } else {
      return defaultValue;
    }
  }
}
```

Se desideri includere la logica di riconnessione nella funzione, consulta Esempio di riconnessione Java.

Esempio di funzione Lambda JavaScript per Amazon Neptune

Note su questo esempio

- Il driver JavaScript non mantiene un pool di connessioni. Apre sempre una singola connessione.
- La funzione di esempio utilizza le utilità di firma SigV4 di gremlin-aws-sigv4 per firmare le richieste a un database abilitato per l'autenticazione IAM.
- Utilizza la funzione retry( ) del modulo di utilità async open source per gestire i tentativi di backoff e nuovo tentativo.
- I passaggi del terminale Gremlin restituiscono un elemento `promise` JavaScript (consulta la documentazione di TinkerPop). Per `next()`, questa è una tupla `{value, done}`.
- Gli errori di connessione vengono generati all'interno del gestore e risolti utilizzando una logica di backoff e nuovo tentativo in linea con le raccomandazioni descritte qui, con un'eccezione. Esiste un tipo di problema di connessione che il driver non considera un'eccezione e che quindi non può essere risolto con questa logica di backoff e nuovo tentativo.

Il problema sta nel fatto che se una connessione viene chiusa dopo che un driver ha inviato una richiesta ma prima che il driver riceva una risposta, la query sembra completata ma restituisce un valore Null. Per quanto riguarda il client della funzione Lambda, la funzione sembra essere stata completata correttamente, ma con una risposta vuota.

L'impatto di questo problema dipende dal modo in cui l'applicazione tratta una risposta vuota. Alcune applicazioni considerano una risposta vuota da una richiesta di lettura come un errore, ma altre potrebbero considerarla erroneamente come un risultato vuoto.

Anche le richieste di scrittura che riscontrano questo problema di connessione restituiranno una risposta vuota. Un'invocazione riuscita con una risposta vuota segnala un'operazione riuscita o un errore? Se il client che richiama una funzione di scrittura considera riuscita un'invocazione della funzione semplicemente se è stato eseguito il commit della scrittura sul database, anziché esaminare il corpo della risposta, è possibile che il sistema perda dati.

Questo problema deriva dal modo in cui il driver tratta gli eventi generati dal socket sottostante. Quando il socket di rete sottostante viene chiuso con un errore `ECONNRESET`, il WebSocket utilizzato dal driver viene chiuso e genera un evento `'ws close'`. Tuttavia, non c'è nulla nel driver che possa gestire quell'evento in un modo che possa essere usato per generare un'eccezione. Di conseguenza, la query semplicemente scompare.

Per risolvere questo problema, la funzione Lambda di esempio qui riportata aggiunge un gestore di eventi `'ws close'` che genera un'eccezione al driver durante la creazione di una connessione remota. Questa eccezione, tuttavia, non viene generata lungo il percorso di richiesta-risposta della query Gremlin e non può quindi essere utilizzata per attivare alcuna logica di backoff e nuovo tentativo all'interno della funzione Lambda stessa. Invece, l'eccezione generata dal gestore di eventi `'ws close'` genera un'eccezione non gestita che causa l'esito negativo dell'invocazione Lambda. Questo consente al client che richiama la funzione di gestire l'errore e di ripetere l'invocazione Lambda, se appropriato.

Ti consigliamo di implementare la logica di backoff e nuovo tentativo nella funzione Lambda stessa per proteggere i client da problemi di connessione intermittenti. Tuttavia, la soluzione alternativa per il problema precedente richiede che il client implementi anche la logica di ripetizione, per gestire gli errori derivanti da questo problema di connessione specifico.

Codice Javascript

```
const gremlin = require('gremlin');
const async = require('async');
const {getUrlAndHeaders} = require('gremlin-aws-sigv4/lib/utils');

const traversal = gremlin.process.AnonymousTraversalSource.traversal;
const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;
const t = gremlin.process.t;
const __ = gremlin.process.statics;

let conn = null;
let g = null;

async function query(context) {

  const id = context.id;

  return g.V(id)
    .fold()
    .coalesce(
      __.unfold(),
      __.addV('User').property(t.id, id)
    )
    .id().next();
}

async function doQuery() {
  const id = Math.floor(Math.random() * 10000).toString();

  let result = await query({id: id});
  return result['value'];
}

exports.handler = async (event, context) => {

  const getConnectionDetails = () => {
    if (process.env['USE_IAM'] == 'true'){
       return getUrlAndHeaders(
         process.env['NEPTUNE_ENDPOINT'],
         process.env['NEPTUNE_PORT'],
         {},
         '/gremlin',
```

```
        'wss');
    } else {
      const database_url = 'wss://' + process.env['NEPTUNE_ENDPOINT'] + ':' +
process.env['NEPTUNE_PORT'] + '/gremlin';
      return { url: database_url, headers: {}};
    }
 };


 const createRemoteConnection = () => {
   const { url, headers } = getConnectionDetails();

   const c = new DriverRemoteConnection(
     url,
     {
       mimeType: 'application/vnd.gremlin-v2.0+json',
       headers: headers
     });

    c._client._connection.on('close', (code, message) => {
        console.info(`close - ${code} ${message}`);
        if (code == 1006){
          console.error('Connection closed prematurely');
          throw new Error('Connection closed prematurely');
        }
      });

    return c;
 };

 const createGraphTraversalSource = (conn) => {
   return traversal().withRemote(conn);
 };

 if (conn == null){
   console.info("Initializing connection")
   conn = createRemoteConnection();
   g = createGraphTraversalSource(conn);
 }

 return async.retry(
   {
     times: 5,
     interval: 1000,
```

```
      errorFilter: function (err) {

        // Add filters here to determine whether error can be retried
        console.warn('Determining whether retriable error: ' + err.message);

        // Check for connection issues
        if (err.message.startsWith('WebSocket is not open')){
          console.warn('Reopening connection');
          conn.close();
          conn = createRemoteConnection();
          g = createGraphTraversalSource(conn);
          return true;
        }

        // Check for ConcurrentModificationException
        if (err.message.includes('ConcurrentModificationException')){
          console.warn('Retrying query because of ConcurrentModificationException');
          return true;
        }

        // Check for ReadOnlyViolationException
        if (err.message.includes('ReadOnlyViolationException')){
          console.warn('Retrying query because of ReadOnlyViolationException');
          return true;
        }

        return false;
      }

    },
    doQuery);
};
```

Esempio di funzione Lambda Python per Amazon Neptune

Ecco alcuni aspetti da notare sulla funzione di esempio AWS Lambda Python seguente:

- Utilizza il modulo backoff.
- Imposta `pool_size=1` per evitare di creare un pool di connessioni non necessario.
- Imposta `message_serializer=serializer.GraphSONSerializersV2d0()`.

```
import os, sys, backoff, math
from random import randint
from gremlin_python import statics
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection
from gremlin_python.driver.protocol import GremlinServerError
from gremlin_python.driver import serializer
from gremlin_python.process.anonymous_traversal import traversal
from gremlin_python.process.graph_traversal import __
from gremlin_python.process.strategies import *
from gremlin_python.process.traversal import T
from aiohttp.client_exceptions import ClientConnectorError
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import ReadOnlyCredentials
from types import SimpleNamespace

import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)


reconnectable_err_msgs = [
    'ReadOnlyViolationException',
    'Server disconnected',
    'Connection refused',
    'Connection was already closed',
    'Connection was closed by server',
    'Failed to connect to server: HTTP Error code 403 - Forbidden'
]

retriable_err_msgs = ['ConcurrentModificationException'] + reconnectable_err_msgs

network_errors = [OSError, ClientConnectorError]

retriable_errors = [GremlinServerError, RuntimeError, Exception] + network_errors

def prepare_iamdb_request(database_url):

    service = 'neptune-db'
    method = 'GET'

    access_key = os.environ['AWS_ACCESS_KEY_ID']
    secret_key = os.environ['AWS_SECRET_ACCESS_KEY']
```

```
    region = os.environ['AWS_REGION']
    session_token = os.environ['AWS_SESSION_TOKEN']

    creds = SimpleNamespace(
        access_key=access_key, secret_key=secret_key, token=session_token,
 region=region,
    )

    request = AWSRequest(method=method, url=database_url, data=None)
    SigV4Auth(creds, service, region).add_auth(request)

    return (database_url, request.headers.items())

def is_retriable_error(e):

    is_retriable = False
    err_msg = str(e)

    if isinstance(e, tuple(network_errors)):
        is_retriable = True
    else:
        is_retriable = any(retriable_err_msg in err_msg for retriable_err_msg in
 retriable_err_msgs)

    logger.error('error: [{}] {}'.format(type(e), err_msg))
    logger.info('is_retriable: {}'.format(is_retriable))

    return is_retriable

def is_non_retriable_error(e):
    return not is_retriable_error(e)

def reset_connection_if_connection_issue(params):

    is_reconnectable = False

    e = sys.exc_info()[1]
    err_msg = str(e)

    if isinstance(e, tuple(network_errors)):
        is_reconnectable = True
    else:
        is_reconnectable = any(reconnectable_err_msg in err_msg for
 reconnectable_err_msg in reconnectable_err_msgs)
```

```
    logger.info('is_reconnectable: {}'.format(is_reconnectable))

    if is_reconnectable:
        global conn
        global g
        conn.close()
        conn = create_remote_connection()
        g = create_graph_traversal_source(conn)

@backoff.on_exception(backoff.constant,
    tuple(retriable_errors),
    max_tries=5,
    jitter=None,
    giveup=is_non_retriable_error,
    on_backoff=reset_connection_if_connection_issue,
    interval=1)
def query(**kwargs):

    id = kwargs['id']

    return (g.V(id)
        .fold()
        .coalesce(
            __.unfold(),
            __.addV('User').property(T.id, id)
        )
        .id().next())

def doQuery(event):
    return query(id=str(randint(0, 10000)))

def lambda_handler(event, context):
    result = doQuery(event)
    logger.info('result - {}'.format(result))
    return result

def create_graph_traversal_source(conn):
    return traversal().withRemote(conn)

def create_remote_connection():
    logger.info('Creating remote connection')

    (database_url, headers) = connection_info()
```

```
    return DriverRemoteConnection(
        database_url,
        'g',
        pool_size=1,
        message_serializer=serializer.GraphSONSerializersV2d0(),
        headers=headers)

def connection_info():

    database_url = 'wss://{}:{}/gremlin'.format(os.environ['neptuneEndpoint'],
 os.environ['neptunePort'])

    if 'USE_IAM' in os.environ and os.environ['USE_IAM'] == 'true':
        return prepare_iamdb_request(database_url)
    else:
        return (database_url, {})

conn = create_remote_connection()
g = create_graph_traversal_source(conn)
```

Ecco alcuni risultati di esempio, che mostrano periodi alternati di carichi pesanti e leggeri:


GremlinWebSocketOpenConnections
Count
261
131
0
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
neptunedbinstance-l22qsahgvgvl
replica-python-lambdas-neptunebasestack-5rzhq9jvp2ri
GremlinRequestsPreSecond
Count/Second
861
430
0
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
neptunedbinstance-l22qsahgvgvl
replica-python-lambdas-neptunebasestack-5rzhq9jvp2ri
Lambda-ConcurrentExecutions
Count
86
43.5
1
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
social-b6501310_add-follows-relationship social-b6501310_add-follows-relationship
social-b6501310_create-user social-b6501310_create-user
social-b6501310_get-recent-follows social-b6501310_get-recent-follows
social-b6501310_get-user-details social-b6501310_get-user-details
Lambda-P99-Latency
Milliseconds
2.10k
1.05k
5.46
06:05 06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
social-b6501310_add-follows-relationship
social-b6501310_create-user
social-b6501310_get-recent-follows
social-b6501310_get-user-details
social-b6501310_recommend-new-follows
Workload
Various units
Count
2.17k
1.09k
9e-3
147k
73.7k
2
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
P99Latency
AverageThroughput
Succeeded
Errors
Count
1
0.5
0
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
social-b6501310_add-follows-relationship
social-b6501310_create-user
social-b6501310_get-recent-follows
social-b6501310_get-user-details
social-b6501310_recommend-new-follows

Amazon Neptune ML per machine learning sui grafi

Spesso in grandi set di dati connessi sono presenti informazioni preziose difficili da estrarre usando query basate solo sull'intuito umano. Le tecniche di machine learning (ML) consentono di trovare correlazioni nascoste nei grafi con miliardi di relazioni. Queste correlazioni possono essere utili per suggerire prodotti, prevedere l'affidabilità creditizia, identificare frodi e molte altre attività.

La funzionalità Neptune ML consente di creare e addestrare modelli di machine learning utili su grafi di grandi dimensioni in poche ore anziché in settimane. A tale scopo, Neptune ML utilizza la tecnologia delle reti neurali a grafo (GNN) basata su Amazon SageMaker e Deep Graph Library (DGL), che è open source. Le reti neurali a grafo sono un campo emergente nell'intelligenza artificiale (vedi, ad esempio, A Comprehensive Survey on Graph Neural Networks). Per un tutorial pratico sull'uso delle reti GNN con la libreria DGL, consulta Learning graph neural networks with Deep Graph Library.

> **Note**
>
> I vertici del grafo nei modelli Neptune ML sono identificati come "nodi". Ad esempio, la classificazione dei vertici utilizza un modello di machine learning per la classificazione dei nodi e la regressione dei vertici utilizza un modello di regressione dei nodi.

Cosa può fare Neptune ML

Neptune supporta sia l'inferenza trasduttiva, che restituisce previsioni precalcolate al momento dell'addestramento, sulla base dei dati del grafo in quel momento, sia l'inferenza induttiva, che restituisce l'elaborazione dei dati e la valutazione del modello in tempo reale, sulla base dei dati correnti. Per informazioni, consultare Differenza tra inferenza induttiva e trasduttiva.

Neptune ML può addestrare modelli di machine learning per supportare cinque diverse categorie di inferenza:

Tipi di attività di inferenza attualmente supportati da Neptune ML

- Classificazione dei nodi: previsione della funzionalità categoriale di una proprietà del vertice.

Ad esempio, per il film Le ali della libertà, Neptune ML può prevedere la sua proprietà `genre` come `story` da un set di candidati di `[story, crime, action, fantasy, drama, family, ...]`.

Esistono due tipi di attività di classificazione dei nodi:

- Classificazione a classe singola: in questo tipo di attività, ogni nodo ha una sola funzionalità di destinazione. Ad esempio, la proprietà `Place_of_birth` di `Alan Turing` ha il valore UK.
- Classificazione multi-classe: in questo tipo di attività, ogni nodo può avere più di una funzionalità di destinazione. Ad esempio, la proprietà `genre` del film Il Padrino ha i valori `crime` e `story`.

- Regressione dei nodi: previsione di una proprietà numerica di un vertice.

  Ad esempio, per il film Avengers: Endgame, Neptune ML può prevedere che la proprietà `popularity` abbia un valore di `5.0`.

- Classificazione degli archi: previsione della funzionalità categoriale di una proprietà dell'arco.

  Esistono due tipi di attività di classificazione degli archi:

  - Classificazione a classe singola: in questo tipo di attività, ogni arco ha una sola funzionalità di destinazione. Ad esempio, un arco delle valutazioni tra un utente e un film potrebbe avere la proprietà `liked`, con un valore "Sì" o "No".
  - Classificazione multi-classe: in questo tipo di attività, ogni arco può avere più di una funzionalità di destinazione. Ad esempio, un arco delle valutazioni tra un utente e un film potrebbe avere più valori per il tag della proprietà come "Divertente" "Commovente", "Agghiacciante" e così via.

- Regressione degli archi: previsione di una proprietà numerica di un arco.

  Ad esempio, un arco delle valutazioni tra un utente e un film potrebbe avere la proprietà numerica `score`, per la quale Neptune ML potrebbe prevedere un valore dato un utente e un film.

- Previsione dei collegamenti: previsione dei nodi di destinazione più probabili per un nodo di origine e un arco in uscita specifici oppure i nodi di origine più probabili per un nodo di destinazione e un arco in entrata specificati.

  Ad esempio, con un grafo della conoscenza per farmaco-malattia, dato `Aspirin` come nodo di origine e `treats` come arco in uscita, Neptune ML può prevedere i nodi di destinazione più pertinenti come `heart disease`, `fever` e così via.

Oppure, con il grafo della conoscenza Wikimedia, dato `President-of` come arco o relazione e `United-States` come nodo di destinazione, Neptune ML può prevedere i presidenti più pertinenti come `George Washington`, `Abraham Lincoln`, `Franklin D. Roosevelt` e così via.

> **Note**
>
> La classificazione dei nodi e la classificazione degli archi supportano solo valori stringa. Ciò significa che i valori delle proprietà numeriche come `0` o `1` non sono supportati, sebbene gli equivalenti stringa `"0"` e `"1"` lo siano. Allo stesso modo, i valori delle proprietà booleane `true` e `false` non sono supportati, ma `"true"` e `"false"` funzionano.

Con Neptune ML, puoi utilizzare modelli di machine learning che rientrano in due categorie generali:

Tipi di modelli di machine learning attualmente supportati da Neptune ML

- Modelli GNN (rete neurale a grafo): includono le reti r-GCN (reti convoluzionali a grafo relazionali). I modelli GNN funzionano per tutti e tre i tipi di attività descritti in precedenza.
- Modelli di incorporamento dei grafi della conoscenza (KGE): includono i modelli `TransE`, `DistMult` e `RotatE`. Funzionano solo per la previsione dei collegamenti.

Modelli definiti dall'utente: Neptune ML consente inoltre di fornire l'implementazione di un modello personalizzato per tutti i tipi di attività sopra elencati. Puoi utilizzare il kit di strumenti per Neptune ML per sviluppare e testare l'implementazione del modello personalizzato basato su Python prima di utilizzare l'API di addestramento Neptune ML con il modello. Consulta Modelli personalizzati in Neptune ML. per informazioni dettagliate su come strutturare e organizzare l'implementazione affinché sia compatibile con l'infrastruttura di addestramento di Neptune ML.

Configurazione di Neptune ML

Il modo più semplice di iniziare a utilizzare Neptune ML è usare il modello di avvio rapido AWS CloudFormation. Questo modello installa tutti i componenti necessari, tra cui un nuovo cluster database Neptune, tutti i ruoli IAM necessari e un nuovo notebook per grafi Neptune per semplificare l'utilizzo di Neptune ML.

Puoi anche installare Neptune ML manualmente, come spiegato in Configurazione di Neptune ML senza utilizzare il modello di avvio rapido AWS CloudFormation.

Utilizzo del modello AWS CloudFormation di Neptune ML per iniziare rapidamente in un nuovo cluster database

Il modo più semplice di iniziare a utilizzare Neptune ML è usare il modello di avvio rapido AWS CloudFormation. Questo modello installa tutti i componenti necessari, tra cui un nuovo cluster database Neptune, tutti i ruoli IAM necessari e un nuovo notebook per grafi Neptune per semplificare l'utilizzo di Neptune ML.

Per creare lo stack di avvio rapido di Neptune ML

1. Per avviare lo stack AWS CloudFormation nella console AWS CloudFormation, scegli uno dei pulsanti Avvia lo stack nella seguente tabella:

Region	Vista	Visualizzazione in Designer	Avvia
Stati Uniti orientali (Virginia settentrionale)	Visualizzazione	Visualizzazione in Designer	
Stati Uniti orientali (Ohio)	Visualizzazione	Visualizzazione in Designer	
Stati Uniti occidentali (California settentrionale)	Visualizzazione	Visualizzazione in Designer	
Stati Uniti occidentali (Oregon)	Visualizzazione	Visualizzazione in Designer	
Canada (Centrale)	Visualizzazione	Visualizzazione in Designer	
Sud America (San Paolo)	Visualizzazione	Visualizzazione in Designer	
Europa (Stoccolma)	Visualizzazione	Visualizzazione in Designer	

Region	Vista	Visualizzazione in Designer	Avvia
Europa (Irlanda)	Visualizzazione	Visualizzazione in Designer	
Europa (Londra)	Visualizzazione	Visualizzazione in Designer	
Europa (Parigi)	Visualizzazione	Visualizzazione in Designer	
Europa (Francoforte)	Visualizzazione	Visualizzazione in Designer	
Medio Oriente (Bahrein)	Visualizzazione	Visualizzazione in Designer	
Medio Oriente (Emirati Arabi Uniti)	Visualizzazione	Visualizzazione in Designer	
Israele (Tel Aviv)	Visualizzazione	Visualizzazione in Designer	
Africa (Città del Capo)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Hong Kong)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Tokyo)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Seul)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Singapore)	Visualizzazione	Visualizzazione in Designer	

Region	Vista	Visualizzazione in Designer	Avvia
Asia Pacifico (Sydney)	Visualizzazione	Visualizzazione in Designer	
Asia Pacifico (Mumbai)	Visualizzazione	Visualizzazione in Designer	
Cina (Pechino)	Visualizzazione	Visualizzazione in Designer	
Cina (Ningxia)	Visualizzazione	Visualizzazione in Designer	
AWS GovCloud (Stati Uniti occidentali)	Visualizzazione	Visualizzazione in Designer	

2. Nella pagina Select Template (Seleziona modello), selezionare Next (Avanti).
3. Nella pagina Specify Details (Specifica dettagli), selezionare Next (Avanti).
4. Nella pagina Opzioni, scegli Avanti.
5. Nella pagina Rivedi sono presenti due caselle di controllo da selezionare:

   - La prima consente di accettare che AWS CloudFormation potrebbe creare risorse IAM con nomi personalizzati.
   - La seconda consente di accettare che AWS CloudFormation potrebbe richiedere la funzionalità `CAPABILITY_AUTO_EXPAND` per il nuovo stack. `CAPABILITY_AUTO_EXPAND` consente esplicitamente a AWS CloudFormation di espandere automaticamente le macro durante la creazione dello stack, senza revisione preliminare.

     Spesso i clienti creano un set di modifiche da un modello elaborato per consentire la revisione delle modifiche apportate dalle macro prima dell'effettiva creazione dello stack. Per ulteriori informazioni, consulta l'API CreateStack di AWS CloudFormation.

   Quindi, scegli Crea.

Il modello di avvio rapido crea e imposta quanto segue:

- Un cluster database Neptune.
- I ruoli IAM necessari (e li associa).
- Il gruppo di sicurezza Amazon EC2 necessario.
- Gli endpoint VPC SageMaker necessari.
- Un gruppo di parametri del cluster database per Neptune ML.
- I parametri necessari in tale gruppo di parametri.
- Un notebook SageMaker con esempi di notebook prepopolati per Neptune ML. Tieni presente che non tutte le dimensioni delle istanze sono disponibili in ogni regione, quindi devi assicurarti che la dimensione dell'istanza notebook selezionata sia supportata dalla tua regione.
- Il servizio Neptune-Export.

Quando lo stack di avvio rapido è pronto, passa al notebook SageMaker creato dal modello e verifica gli esempi prepopolati. Ti aiuteranno a scaricare set di dati di esempio da utilizzare per sperimentare le funzionalità di Neptune ML.

Possono anche farti risparmiare molto tempo durante l'utilizzo di Neptune ML. Ad esempio, guarda il comando magic di riga %neptune_ml e il comando magic di cella %%neptune_ml supportati da questi notebook.

Puoi inoltre utilizzare il parametro AWS CLI seguente per eseguire il modello di avvio rapido AWS CloudFormation:

```
aws cloudformation create-stack \
  --stack-name neptune-ml-fullstack-$(date '+%Y-%m-%d-%H-%M') \
  --template-url https://aws-neptune-customer-samples.s3.amazonaws.com/v2/
cloudformation-templates/neptune-ml-nested-stack.json \
  --parameters ParameterKey=EnableIAMAuthOnExportAPI,ParameterValue=(true if you have
 IAM auth enabled, or false otherwise) \
               ParameterKey=Env,ParameterValue=test$(date '+%H%M')\
  --capabilities CAPABILITY_IAM \
  --region (the AWS region, like us-east-1) \
  --disable-rollback \
  --profile (optionally, a named CLI profile of yours)
```

Configurazione di Neptune ML senza utilizzare il modello di avvio rapido AWS CloudFormation

1. Inizia con un cluster database Neptune funzionante

Se non utilizzi il modello di avvio rapido AWS CloudFormation per configurare Neptune ML, dovrai usare un cluster database Neptune esistente. Se lo desideri, puoi usarne uno che hai già o clonarne uno che usi già oppure puoi crearne uno nuovo (consulta Creazione di un cluster DB).

2. Installazione del servizio Neptune-Export

Se non hai ancora provveduto, installa il servizio Neptune-Export, come spiegato in Utilizzo del servizio Neptune-Export per esportare i dati di Neptune.

Aggiungi una regola in entrata al gruppo di sicurezza `NeptuneExportSecurityGroup` creato dal programma di installazione, con le seguenti impostazioni:

- Tipo: `Custom TCP`
- Protocol (Protocollo): `TCP`
- Intervallo porte: `80 - 443`
- Origine: *`(ID del gruppo di sicurezza del cluster database Neptune`*)

3. Crea un ruolo IAM **`NeptuneLoadFromS3`** personalizzato

Se ancora non hai provveduto, crea un ruolo IAM `NeptuneLoadFromS3` personalizzato, come spiegato in Creazione di un ruolo IAM per l'accesso ad Amazon S3.

Per creare un ruolo **`NeptuneSageMakerIAMRole`** personalizzato

Utilizza la console IAM per creare un ruolo `NeptuneSageMakerIAMRole` personalizzato, usando la seguente policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:CreateNetworkInterfacePermission",
        "ec2:CreateVpcEndpoint",
```

```
        "ec2:DeleteNetworkInterface",
        "ec2:DeleteNetworkInterfacePermission",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/*"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "sagemaker.amazonaws.com"
          ]
        }
      },
      "Effect": "Allow"
    },
    {
      "Action": [
        "kms:CreateGrant",
        "kms:Decrypt",
        "kms:GenerateDataKey*"
```

```
      ],
      "Resource": "arn:aws:kms:*:*:key/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/sagemaker/*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:AddTags",
        "sagemaker:CreateEndpoint",
        "sagemaker:CreateEndpointConfig",
        "sagemaker:CreateHyperParameterTuningJob",
        "sagemaker:CreateModel",
        "sagemaker:CreateProcessingJob",
        "sagemaker:CreateTrainingJob",
        "sagemaker:CreateTransformJob",
        "sagemaker:DeleteEndpoint",
        "sagemaker:DeleteEndpointConfig",
        "sagemaker:DeleteModel",
        "sagemaker:DescribeEndpoint",
        "sagemaker:DescribeEndpointConfig",
        "sagemaker:DescribeHyperParameterTuningJob",
        "sagemaker:DescribeModel",
        "sagemaker:DescribeProcessingJob",
        "sagemaker:DescribeTrainingJob",
        "sagemaker:DescribeTransformJob",
        "sagemaker:InvokeEndpoint",
        "sagemaker:ListTags",
        "sagemaker:ListTrainingJobsForHyperParameterTuningJob",
        "sagemaker:StopHyperParameterTuningJob",
        "sagemaker:StopProcessingJob",
        "sagemaker:StopTrainingJob",
```

```
        "sagemaker:StopTransformJob",
        "sagemaker:UpdateEndpoint",
        "sagemaker:UpdateEndpointWeightsAndCapacities"
      ],
      "Resource": [
        "arn:aws:sagemaker:*:*:*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:ListEndpointConfigs",
        "sagemaker:ListEndpoints",
        "sagemaker:ListHyperParameterTuningJobs",
        "sagemaker:ListModels",
        "sagemaker:ListProcessingJobs",
        "sagemaker:ListTrainingJobs",
        "sagemaker:ListTransformJobs"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:AbortMultipartUpload",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Durante la creazione di questo ruolo, modifica la relazione di trust affinché corrisponda a quella seguente:

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "ec2.amazonaws.com",
          "rds.amazonaws.com",
          "sagemaker.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Infine, copia il nome ARN assegnato a questo nuovo ruolo `NeptuneSageMakerIAMRole`.

> **Important**
>
> - Assicurati che le autorizzazioni Amazon S3 indicate in `NeptuneSageMakerIAMRole` corrispondano a quelle precedenti.
> - Il nome ARN universale, `arn:aws:s3:::*`, è usato per la risorsa Amazon S3 nella policy precedente. Se per qualche motivo non è possibile utilizzare il nome ARN universale, è necessario aggiungere `arn:aws:s3:::graphlytics*` e il nome ARN di qualsiasi altra risorsa Amazon S3 del cliente che i comandi Neptune ML utilizzeranno nella sezione delle risorse.

Configurazione del cluster database per abilitare Neptune ML

Per configurare il cluster database per Neptune ML

1. Nella console Neptune passa a Gruppi di parametri e quindi al gruppo di parametri del cluster database associato al cluster database che intendi usare. Imposta il parametro `neptune_ml_iam_role` sul nome ARN assegnato al ruolo `NeptuneSageMakerIAMRole` appena creato.
2. Passa a Database, quindi seleziona il cluster database che utilizzerai per Neptune ML. Seleziona Azioni, quindi Gestisci ruoli IAM.

3. Nella pagina Gestisci ruoli IAM seleziona Aggiungi ruolo e aggiungi `NeptuneSageMakerIAMRole`. Aggiungi quindi il ruolo `NeptuneLoadFromS3`.
4. Riavvia l'istanza di scrittura del cluster database.

Creazione di due endpoint SageMaker nel VPC Neptune

Infine, per consentire al motore Neptune di accedere alle API di gestione SageMaker necessarie, occorre creare due endpoint SageMaker nel VPC Neptune, come spiegato in Creazione di due endpoint per SageMaker nel VPC Neptune.

Configurazione manuale di un notebook Neptune per Neptune ML

I notebook Neptune SageMaker sono precaricati con una varietà di notebook di esempio per Neptune ML. Puoi visualizzare in anteprima questi esempi nel repository GitHub di notebook per grafi open source.

Puoi usare uno dei notebook Neptune esistenti o, se preferisci, puoi crearne uno personalizzato, seguendo le istruzioni riportate in Utilizzo Neptune Workbench per ospitare i notebook Neptune.

Puoi anche configurare un notebook Neptune predefinito da utilizzare con Neptune ML seguendo questa procedura:

Modifica di un notebook per Neptune ML

1. Apri la console di Amazon SageMaker all'indirizzo https://console.aws.amazon.com/sagemaker/.
2. Nel riquadro di navigazione a sinistra scegli Notebook, quindi Istanze notebook. Cerca il nome del notebook Neptune da usare per Neptune ML e selezionalo per accedere alla pagina dei dettagli.
3. Se l'istanza notebook è in esecuzione, seleziona il pulsante Arresta in alto a destra nella pagina dei dettagli del notebook.
4. In Impostazioni dell'istanza Notebook, in Configurazione del ciclo di vita seleziona il collegamento per aprire la pagina relativa al ciclo di vita del notebook.
5. Seleziona Modifica in alto a destra, quindi fai clic su Continua.
6. Nella scheda Avvia notebook modifica lo script per includere comandi di esportazione aggiuntivi e per compilare i campi per il ruolo IAM di Neptune ML e l'URI del servizio di esportazione. L'aspetto sarà simile al seguente a seconda della shell (interprete di comandi):

```
echo "export NEPTUNE_ML_ROLE_ARN=(your Neptune ML IAM role ARN)" >> ~/.bashrc
echo "export NEPTUNE_EXPORT_API_URI=(your export service URI)" >> ~/.bashrc
```

7. Selezionare Update (Aggiorna).
8. Torna alla pagina dell'istanza notebook. In Autorizzazioni e crittografia è disponibile un campo per ARN del ruolo IAM. Seleziona il collegamento in questo campo per passare al ruolo IAM con cui viene eseguita questa istanza notebook.
9. Crea una nuova policy inline come questa:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "arn:aws:cloudwatch:[AWS_REGION]:[AWS_ACCOUNT_ID]:*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "arn:aws:logs:[AWS_REGION]:[AWS_ACCOUNT_ID]:*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:Put*",
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "arn:aws:s3:::*",
      "Effect": "Allow"
    },
    {
      "Action": "execute-api:Invoke",
```

```
      "Resource": "arn:aws:execute-api:[AWS_REGION]:[AWS_ACCOUNT_ID]:*/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:CreateModel",
        "sagemaker:CreateEndpointConfig",
        "sagemaker:CreateEndpoint",
        "sagemaker:DescribeModel",
        "sagemaker:DescribeEndpointConfig",
        "sagemaker:DescribeEndpoint",
        "sagemaker:DeleteModel",
        "sagemaker:DeleteEndpointConfig",
        "sagemaker:DeleteEndpoint"
      ],
      "Resource": "arn:aws:sagemaker:[AWS_REGION]:[AWS_ACCOUNT_ID]:*/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "[YOUR_NEPTUNE_ML_IAM_ROLE_ARN]"",
      "Effect": "Allow"
    }
  ]
}
```

10. Salva questa nuova policy e collegala al ruolo IAM nel passaggio 8.
11. Seleziona Avvia in alto a destra nella pagina dei dettagli dell'istanza notebook SageMaker per avviare l'istanza notebook.

Utilizzo della AWS CLI per configurare Neptune ML in un cluster database

Oltre al modello di avvio rapido AWS CloudFormation e alla console AWS Management Console, puoi anche configurare Neptune ML utilizzando la AWS CLI.

1. Scegli un gruppo di parametri del cluster database per il nuovo cluster Neptune ML

I comandi AWS CLI seguenti creano un nuovo gruppo di parametri del cluster database e lo configurano per l'utilizzo con Neptune ML:

Per creare e configurare un gruppo di parametri del cluster database per Neptune ML

1. Crea un nuovo gruppo di parametri del cluster database:

```
aws neptune create-db-cluster-parameter-group \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --db-parameter-group-family neptune1
  --description "(description of your machine learning project)" \
  --region (AWS region, such as us-east-1)
```

2. Crea un parametro `neptune_ml_iam_role` del cluster database impostato sul nome ARN del `SageMakerExcecutionIAMRole` che verrà usato dal cluster database durante la chiamata a SageMaker per la creazione di processi e il recupero di previsioni dai modelli ML ospitati:

```
aws neptune modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --parameters "ParameterName=neptune_ml_iam_role, \
                ParameterValue=ARN of the SageMakerExcecutionIAMRole, \
                Description=NeptuneMLRole, \
                ApplyMethod=pending-reboot" \
  --region (AWS region, such as us-east-1)
```

   L'impostazione di questo parametro consente a Neptune di accedere a SageMaker senza dover passare il ruolo con ogni chiamata.

   Per informazioni su come creare il ruolo `SageMakerExcecutionIAMRole`, consulta Per creare un ruolo `NeptuneSageMakerIAMRole` personalizzato.

3. Infine, usa `describe-db-cluster-parameters` per verificare che tutti i parametri nel nuovo gruppo di parametri del cluster database siano impostati come desiderato:

```
aws neptune describe-db-cluster-parameters \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --region (AWS region, such as us-east-1)
```

Collegamento del nuovo gruppo di parametri del cluster database al cluster database verrà usato con Neptune ML

Ora puoi collegare il nuovo gruppo di parametri del cluster database appena creato a un cluster database esistente utilizzando il seguente comando:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (the name of your existing DB cluster) \
  --apply-immediately
  --db-cluster-parameter-group-name (name of your new DB cluster parameter group) \
  --region (AWS region, such as us-east-1)
```

Per rendere effettivi tutti i parametri, puoi quindi riavviare il cluster database:

```
aws neptune reboot-db-instance
  --db-instance-identifier (name of the primary instance of your DB cluster) \
  --profile (name of your AWS profile to use) \
  --region (AWS region, such as us-east-1)
```

In alternativa se stai creando un nuovo cluster database da utilizzare con Neptune ML, puoi usare il seguente comando per creare il cluster con il nuovo gruppo di parametri collegato, quindi creare una nuova istanza primaria (scrittura):

```
cluster-name=(the name of the new DB cluster)
aws neptune create-db-cluster
  --db-cluster-identifier ${cluster-name}
  --engine graphdb \
  --engine-version 1.0.4.1 \
  --db-cluster-parameter-group-name (name of your new DB cluster parameter group) \
  --db-subnet-group-name (name of the subnet to use) \
  --region (AWS region, such as us-east-1)

aws neptune create-db-instance
  --db-cluster-identifier ${cluster-name}
  --db-instance-identifier ${cluster-name}-i \
```

```
  --db-instance-class (the instance class to use, such as db.r5.xlarge)
  --engine graphdb \
  --region (AWS region, such as us-east-1)
```

Collegamento di **NeptuneSageMakerIAMRole** al cluster database in modo che possa accedere alle risorse SageMaker e Amazon S3

Infine, segui le istruzioni in Per creare un ruolo `NeptuneSageMakerIAMRole` personalizzato per creare un ruolo IAM che consenta al tuo cluster database di comunicare con SageMaker e Amazon S3. Quindi, usa il seguente comando per associare il ruolo `NeptuneSageMakerIAMRole` che hai creato al cluster database:

```
aws neptune add-role-to-db-cluster
  --db-cluster-identifier ${cluster-name}
  --role-arn arn:aws:iam::(the ARN number of the role's ARN):role/NeptuneMLRole \
  --region (AWS region, such as us-east-1)
```

Creazione di due endpoint per SageMaker nel VPC Neptune

Neptune ML necessita di due endpoint SageMaker nel VPC del cluster database Neptune:

- `com.amazonaws.(AWS region, like us-east-1).sagemaker.runtime`
- `com.amazonaws.(AWS region, like us-east-1).sagemaker.api`

Se non hai utilizzato il modello di avvio rapido AWS CloudFormation, che li crea automaticamente, puoi utilizzare i comandi AWS CLI seguenti per crearli:

Questo crea l'endpoint `sagemaker.runtime`:

```
create-vpc-endpoint
  --vpc-id (the ID of your Neptune DB cluster's VPC)
  --service-name com.amazonaws.(AWS region, like us-east-1).sagemaker.runtime
  --subnet-ids (the subnet ID or IDs that you want to use)
  --security-group-ids (the security group for the endpoint network interface, or omit
 to use the default)
  --private-dns-enabled
```

E questo crea l'endpoint `sagemaker.api`:

```
aws create-vpc-endpoint
```

```
  --vpc-id (the ID of your Neptune DB cluster's VPC)
  --service-name com.amazonaws.(AWS region, like us-east-1).sagemaker.api
  --subnet-ids (the subnet ID or IDs that you want to use)
  --security-group-ids (the security group for the endpoint network interface, or omit
 to use the default)
  --private-dns-enabled
```

Per creare questi endpoint puoi anche utilizzare la console VPC. Consulta anche Secure prediction calls in Amazon SageMaker with AWS PrivateLink e Securing all Amazon SageMaker API calls with AWS PrivateLink.

Creazione di un parametro dell'endpoint di inferenza SageMaker nel gruppo di parametri del cluster database

Per evitare di dover specificare l'endpoint di inferenza SageMaker del modello che stai utilizzando in ogni query che esegui su di esso, crea un parametro del cluster database denominato `neptune_ml_endpoint` nel gruppo di parametri del cluster database per Neptune ML. Imposta il parametro sull'`id` dell'endpoint dell'istanza in questione.

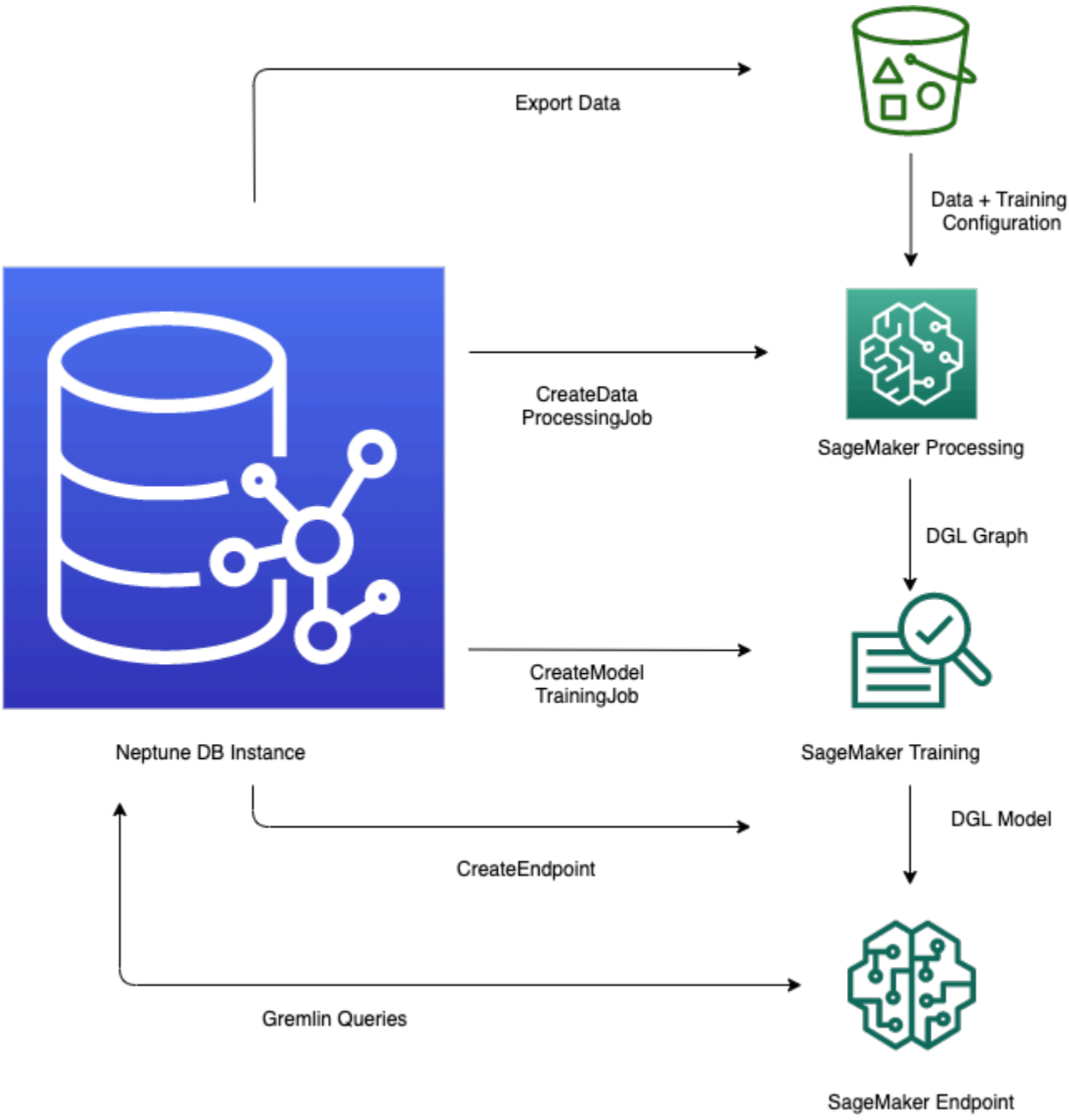
A tale scopo, puoi usare il comando AWS CLI seguente:

```
aws neptune modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name neptune-ml-demo \
  --parameters "ParameterName=neptune_ml_endpoint, \
                ParameterValue=(the name of the SageMaker inference endpoint you want
 to query), \
                Description=NeptuneMLEndpoint, \
                ApplyMethod=pending-reboot" \
  --region (AWS region, such as us-east-1)
```

Panoramica dell'utilizzo della funzionalità Neptune ML

Avvio del flusso di lavoro per l'utilizzo di Neptune ML

Per iniziare a usare la funzionalità Neptune ML in Amazon Neptune sono in genere necessari i cinque passaggi seguenti:



1. Esportazione e configurazione dei dati: la fase di esportazione dei dati utilizza il servizio Neptune-Export o lo strumento da riga di comando `neptune-export` per esportare i dati da Neptune

ad Amazon Simple Storage Service (Amazon S3) in formato CSV. Contemporaneamente viene generato automaticamente un file di configurazione denominato `training-data-configuration.json`, che specifica come è possibile caricare i dati esportati in un grafo addestrabile.

2. Pre-elaborazione dei dati: in questo passaggio, il set di dati esportato viene pre-elaborato utilizzando le tecniche standard per prepararlo all'addestramento del modello. È possibile eseguire la normalizzazione delle funzionalità per i dati numerici e codificare le funzionalità del testo con `word2vec`. Alla fine di questo passaggio, viene generato un grafo DGL (Deep Graph library) dal set di dati esportato che verrà usato per il passaggio di addestramento del modello.

   Questo passaggio viene implementato mediante un processo di elaborazione SageMaker nel tuo account e i dati risultanti vengono archiviati nella posizione Amazon S3 specificata.

3. Addestramento del modello: il passaggio di addestramento del modello addestra il modello di machine learning che verrà utilizzato per le previsioni.

   L'addestramento del modello viene svolto in due fasi:

   - La prima fase utilizza un processo di elaborazione SageMaker per generare un set di configurazione della strategia di addestramento del modello che specifica il tipo di modello e gli intervalli di iperparametri del modello che verranno utilizzati per l'addestramento del modello.
   - La seconda fase utilizza quindi un processo di ottimizzazione del modello SageMaker per provare diverse configurazioni di iperparametri e selezionare il processo di addestramento che ha generato il modello con le migliori prestazioni. Il processo di ottimizzazione esegue un numero prestabilito di prove del processo di addestramento del modello sui dati elaborati. Al termine di questa fase, i parametri del modello addestrato del miglior processo di addestramento vengono utilizzati per generare gli artefatti del modello per l'inferenza.

4. Creazione di un endpoint di inferenza in Amazon SageMaker: l'endpoint di inferenza è un'istanza dell'endpoint SageMaker che viene avviata con gli artefatti del modello generati dal miglior processo di addestramento. Ogni modello è legato a un singolo endpoint. L'endpoint può accettare le richieste in entrata dal database a grafo e restituire le previsioni del modello per gli input nelle richieste. Dopo aver creato l'endpoint, questo rimane attivo finché non viene eliminato.

5. Esecuzione di query sul modello di machine learning con Gremlin: puoi utilizzare le estensioni del linguaggio di query Gremlin per eseguire query sulle previsioni dall'endpoint di inferenza.

> **Note**
>
> Neptune Workbench contiene un comando magic di riga e un comando magic di cella che ti permettono di risparmiare molto tempo nella gestione di questi passaggi, ad esempio:
>
> - %neptune_ml
> - %%neptune_ml

Generazione di previsioni basate sui dati dei grafi in evoluzione

Con un grafo in continua evoluzione, potrebbe essere necessario creare periodicamente nuove previsioni in batch utilizzando dati aggiornati. L'esecuzione di query su previsioni precalcolate (inferenza trasduttiva) può essere significativamente più veloce rispetto alla generazione immediata di nuove previsioni basate sui dati più recenti (inferenza induttiva). Entrambi gli approcci sono validi, a seconda della rapidità con cui cambiano i dati e dei requisiti in termini di prestazioni.

Differenza tra inferenza induttiva e trasduttiva

Quando esegue l'inferenza trasduttiva, Neptune cerca e restituisce previsioni precalcolate al momento dell'addestramento.

Quando esegue l'inferenza induttiva, Neptune costruisce il sottografo pertinente e ne recupera le proprietà. Il modello GNN DGL applica quindi l'elaborazione dei dati e la valutazione del modello in tempo reale.

L'inferenza induttiva può quindi generare previsioni che coinvolgono nodi e archi che non erano presenti al momento dell'addestramento e che riflettono lo stato attuale del grafo. Ciò determina, tuttavia, una maggiore latenza.

Se il grafo è dinamico, è consigliabile utilizzare l'inferenza induttiva per essere sicuri di tenere conto dei dati più recenti, ma se il grafo è statico, l'inferenza trasduttiva è più veloce ed efficiente.

Per impostazione predefinita, l'inferenza induttiva è disabilitata. È possibile abilitarla per eseguire una query utilizzando il predicato Neptune#ml.inductiveInference Gremlin nella query come indicato di seguito:

```
.with( "Neptune#ml.inductiveInference")
```

Flussi di lavoro trasduttivi incrementali

Mentre per aggiornare gli artefatti del modello è sufficiente eseguire nuovamente i passaggi da uno a tre (dall'esportazione e configurazione dei dati alla trasformazione del modello), Neptune ML supporta modi più semplici per aggiornare le previsioni di ML in batch utilizzando nuovi dati. Uno prevede l'utilizzo di un flusso di lavoro basato su modello incrementale e un altro prevede l'utilizzo del riaddestramento del modello con un avvio a caldo.

Flusso di lavoro basato su modello incrementale

In questo flusso di lavoro, si aggiornano le previsioni di ML senza riaddestrare il modello ML.

> **Note**
>
> È possibile farlo solo quando i dati del grafo sono stati aggiornati con nuovi nodi e/o archi. Non funzionerà quando i nodi vengono rimossi.

1. Esportazione e configurazione dei dati: questo passaggio è uguale a quello del flusso di lavoro principale.
2. Pre-elaborazione incrementale dei dati: questo passaggio è simile al passaggio di pre-elaborazione dei dati nel flusso di lavoro principale, ma utilizza la stessa configurazione di elaborazione utilizzata in precedenza, che corrisponde a un modello addestrato specifico.
3. Trasformazione del modello: anziché un passaggio di addestramento del modello, questo passaggio di trasformazione del modello recupera il modello addestrato dal flusso di lavoro principale e dai risultati del passaggio di pre-elaborazione incrementale dei dati e genera nuovi artefatti del modello da utilizzare per l'inferenza. Il passaggio di trasformazione del modello avvia un processo di elaborazione SageMaker per eseguire il calcolo che genera gli artefatti del modello aggiornati.
4. Aggiornamento dell'endpoint di inferenza di Amazon SageMaker: facoltativamente, se è presente un endpoint di inferenza esistente, questo passaggio aggiorna l'endpoint con i nuovi artefatti del modello generati dalla fase di trasformazione del modello. In alternativa, puoi anche creare un nuovo endpoint di inferenza con i nuovi artefatti del modello.

Riaddestramento del modello con avvio a caldo

Utilizzando questo flusso di lavoro, puoi addestrare e implementare un nuovo modello di machine learning per generare previsioni utilizzando i dati del grafo incrementale, ma partire da un modello esistente generato tramite il flusso di lavoro principale:

1. Esportazione e configurazione dei dati: questo passaggio è uguale a quello del flusso di lavoro principale.
2. Pre-elaborazione incrementale dei dati: questo passaggio è uguale a quello del flusso di lavoro di inferenza del modello incrementale. I nuovi dati del grafo devono essere elaborati con lo stesso metodo di elaborazione usato in precedenza per l'addestramento del modello.
3. Addestramento dei modelli con avvio a caldo: l'addestramento dei modelli è simile a quello che avviene nel flusso di lavoro principale, ma è possibile accelerare la ricerca degli iperparametri del modello sfruttando le informazioni dell'attività di addestramento del modello precedente.
4. Aggiornamento dell'endpoint di inferenza Amazon SageMaker: questo passaggio è lo stesso del flusso di lavoro di inferenza del modello incrementale.

Flussi di lavoro per modelli personalizzati in Neptune ML

Neptune ML consente di implementare, addestrare e distribuire modelli personalizzati per qualsiasi attività supportata da Neptune ML. Il flusso di lavoro per lo sviluppo e l'implementazione di un modello personalizzato è essenzialmente lo stesso di quello dei modelli predefiniti, con alcune differenze, come spiegato in Flusso di lavoro per i modelli personalizzati.

Selezione delle istanze per le fasi di Neptune ML

Le diverse fasi dell'elaborazione di Neptune ML utilizzano istanze SageMaker differenti. In questa sezione viene illustrato come scegliere il tipo di istanza corretto per ogni fase. Puoi trovare informazioni sui tipi di istanze e sui prezzi di SageMaker nella pagina dei prezzi di Amazon SageMaker.

Selezione di un'istanza per l'elaborazione dei dati

La fase di elaborazione dei dati di SageMaker richiede un'istanza di elaborazione con memoria e spazio di archiviazione su disco sufficienti per i dati di input, intermedi e di output. La quantità specifica di memoria e spazio di archiviazione su disco necessaria dipende dalle caratteristiche del grafo Neptune ML e dalle relative funzionalità esportate.

Per impostazione predefinita, Neptune ML sceglie l'istanza `ml.r5` più piccola, la cui memoria è dieci volte più grande delle dimensioni dei dati del grafo esportati su disco.

Selezione di un'istanza per l'addestramento e la trasformazione del modello

La scelta del tipo di istanza corretto per l'addestramento del modello o la trasformazione del modello dipende dal tipo di attività, dalle dimensioni del grafo e dai requisiti dei tempi di risposta. Le istanze GPU offrono le prestazioni migliori. In genere è consigliabile usare le istanze seriali `p3` e `g4dn`. Puoi anche usare le istanze `p2` o `p4d`.

Per impostazione predefinita, Neptune ML sceglie l'istanza GPU più piccola con più memoria di quella richiesta dall'addestramento e dalla trasformazione del modello. Puoi trovare la selezione nel file `train_instance_recommendation.json`, nella posizione di output dell'elaborazione dati di Amazon S3. Ecco un esempio del contenuto di un file `train_instance_recommendation.json`:

```
{
  "instance":     "(the recommended instance type for model training and transform)",
  "cpu_instance": "(the recommended instance type for base processing instance)",
  "disk_size":    "(the estimated disk space required)",
  "mem_size":     "(the estimated memory required)"
}
```

Selezione di un'istanza per un endpoint di inferenza

La selezione del tipo di istanza corretto per un endpoint di inferenza dipende dal tipo di attività, dalle dimensioni del grafo e dal budget. Per impostazione predefinita, Neptune ML sceglie l'istanza `ml.m5d` più piccola con più memoria di quella richiesta dall'endpoint di inferenza.

> **Note**
>
> Se sono necessari più di 384 GB di memoria, Neptune ML utilizza un'istanza `ml.r5d.24xlarge`.

Puoi vedere il tipo di istanza consigliato da Neptune ML nel file `infer_instance_recommendation.json` disponibile nella posizione Amazon S3 in uso per l'addestramento del modello. Ecco un esempio del contenuto del file:

```
{
  "instance" :   "(the recommended instance type for an inference endpoint)",
  "disk_size" :  "(the estimated disk space required)",
  "mem_size" :   "(the estimated memory required)"
}
```

Utilizzo dello strumento **`neptune-export`** o del servizio Neptune-Export per esportare dati da Neptune per Neptune ML

Neptune ML richiede di fornire i dati di addestramento per Deep Graph Library (DGL) per la creazione e la valutazione dei modelli.

È possibile esportare dati da Neptune utilizzando il servizio Neptune-Export o l'utilità `neptune-export`. Sia il servizio che lo strumento da riga di comando pubblicano i dati su Amazon Simple Storage Service (Amazon S3) in formato CSV e crittografati tramite la crittografia lato server di Amazon S3 (`SSE-S3`). Per informazioni, consultare File esportati da Neptune-Export e `neptune-export`.

Inoltre, quando si configura un'esportazione dei dati di addestramento per Neptune ML, il processo di esportazione crea e pubblica un file di configurazione crittografato per l'addestramento del modello insieme ai dati esportati. Per impostazione predefinita, questo file è denominato `training-data-configuration.json`.

Esempi di utilizzo del servizio Neptune-Export per esportare i dati di addestramento per Neptune ML

Questa richiesta esporta i dati di addestramento del grafo delle proprietà per un'attività di classificazione dei nodi:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-pg",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": {
          "endpoint": "(your Neptune endpoint DNS name)",
          "profile": "neptune_ml"
        },
        "additionalParams": {
          "neptune_ml": {
            "version": "v2.0",
            "targets": [
              {
                "node": "Movie",
```

```
                "property": "genre",
                "type": "classification"
              }
            ]
          }
        }
      }'
```

Questa richiesta esporta i dati di addestramento RDF per un'attività di classificazione dei nodi:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-rdf",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": {
          "endpoint": "(your Neptune endpoint DNS name)",
          "profile": "neptune_ml"
        },
        "additionalParams": {
          "neptune_ml": {
            "version": "v2.0",
            "targets": [
              {
                "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
                "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/
genre",
                "type": "classification"
              }
            ]
          }
        }
      }'
```

Campi da impostare nell'oggetto **`params`** durante l'esportazione dei dati di addestramento

L'oggetto `params` in una richiesta di esportazione può contenere vari campi, come descritto nella documentazione di `params`. I campi seguenti sono quelli più importanti per l'esportazione dei dati di addestramento di machine learning:

- **`endpoint`**: usa `endpoint` per specificare un endpoint di un'istanza Neptune nel cluster database su cui il processo di esportazione può eseguire query per estrarre i dati.
- **`profile`**: il campo `profile` nell'oggetto `params` deve essere impostato su **`neptune-ml`**.

  In questo modo, il processo di esportazione formatta i dati esportati nel modo appropriato per l'addestramento del modello Neptune ML, in formato CSV per i dati dei grafi di proprietà o come N-Triple per i dati RDF. Inoltre, crea e scrive un file `training-data-configuration.json` nella stessa posizione Amazon S3 dei dati di addestramento esportati.
- **`cloneCluster`**: se impostato su `true`, il processo di esportazione clona il cluster database, esporta dal clone e quindi elimina il clone al termine dell'operazione.
- **`useIamAuth`**: se nel cluster database è abilitata l'autenticazione IAM, è necessario includere questo campo impostato su `true`.

Il processo di esportazione offre anche diversi modi per filtrare i dati esportati (vedi questi esempi).

Utilizzo dell'oggetto **`additionalParams`** per ottimizzare l'esportazione delle informazioni di addestramento dei modelli

L'oggetto `additionalParams` contiene campi che è possibile utilizzare per specificare le etichette e le funzionalità delle classi di machine learning ai fini dell'addestramento e fornire supporto durante la creazione di un file di configurazione dei dati di addestramento.

Il processo di esportazione non può dedurre automaticamente le proprietà dei nodi e degli archi che costituiscono le etichette delle classi di machine learning da usare come esempi ai fini dell'addestramento. Inoltre, non è in grado di dedurre automaticamente la migliore codifica delle funzionalità per le proprietà numeriche, categoriali e di testo, quindi è necessario fornire suggerimenti utilizzando i campi nell'oggetto `additionalParams` per specificare questi dettagli o per sovrascrivere la codifica predefinita.

Per i dati dei grafi di proprietà, la struttura di primo livello di `additionalParams` in una richiesta di esportazione avrà un aspetto simile al seguente:

```
{
  "command": "export-pg",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
```

```
    "profile": "neptune_ml"
  },
  "additionalParams": {
      "neptune_ml": {
        "version": "v2.0",
        "targets": [ (an array of node and edge class label targets) ],
        "features": [ (an array of node feature hints) ]
    }
  }
}
```

Per i dati RDF, la struttura di primo livello avrà un aspetto simile al seguente:

```
{
  "command": "export-rdf",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams": {
      "neptune_ml": {
        "version": "v2.0",
        "targets": [ (an array of node and edge class label targets) ]
    }
  }
}
```

Puoi anche fornire più configurazioni di esportazione, utilizzando il campo `jobs`:

```
{
  "command": "export-pg",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams" : {
    "neptune_ml" : {
      "version": "v2.0",
      "jobs": [
        {
          "name" : "(training data configuration name)",
```

```
          "targets": [ (an array of node and edge class label targets) ],
          "features": [ (an array of node feature hints) ]
        },
        {
          "name" : "(another training data configuration name)",
          "targets": [ (an array of node and edge class label targets) ],
          "features": [ (an array of node feature hints) ]
        }
      ]
    }
  }
}
```

Elementi di primo livello nel campo **neptune_ml** in **additionalParams**

Elemento **version** in **neptune_ml**

Specifica la versione della configurazione dei dati di addestramento da generare.

(Facoltativo), Tipo: stringa, Impostazione predefinita: "v2.0".

Se includi `version`, deve essere impostato su `v2.0`.

Campo **jobs** in **neptune_ml**

Contiene un array di oggetti di configurazione dei dati di addestramento, ognuno dei quali definisce un processo di elaborazione dei dati e contiene:

- **name**: nome della configurazione dei dati di addestramento da creare.

  Ad esempio, una configurazione dei dati di addestramento con il nome "job-number-1" genera un file di configurazione dei dati di addestramento denominato `job-number-1.json`.
- **targets**: array JSON di destinazioni di etichette delle classi dei nodi e degli archi che rappresentano le etichette delle classi di machine learning ai fini dell'addestramento. Per informazioni, consultare Campo `targets` in un oggetto `neptune_ml`.
- **features**: array JSON di funzionalità delle proprietà dei nodi. Per informazioni, consultare Campo `features` in `neptune_ml`.

Campo **`targets`** in un oggetto **`neptune_ml`**

Il campo `targets` in una configurazione di esportazione dei dati di addestramento JSON contiene un array di oggetti di destinazione che specificano un'attività di addestramento e le etichette della classe machine-learning per l'addestramento di questa attività. Il contenuto degli oggetti di destinazione varia a seconda che l'addestramento venga eseguito su dati del grafo delle proprietà o su dati RDF.

Per le attività di classificazione e regressione dei nodi del grafo delle proprietà, gli oggetti di destinazione nell'array avranno l'aspetto seguente:

```
{
  "node": "(node property-graph label)",
  "property": "(property name)",
  "type" : "(used to specify classification or regression)",
  "split_rate": [0.8,0.2,0.0],
  "separator": ","
}
```

Per le attività di classificazione, regressione o previsione dei collegamenti degli archi del grafo delle proprietà, gli oggetti di destinazione nell'array avranno l'aspetto seguente:

```
{
  "edge": "(edge property-graph label)",
  "property": "(property name)",
  "type" : "(used to specify classification, regression or link_prediction)",
  "split_rate": [0.8,0.2,0.0],
  "separator": ","
}
```

Per le attività di classificazione e regressione RDF, gli oggetti di destinazione nell'array avranno l'aspetto seguente:

```
{
  "node": "(node type of an RDF node)",
  "predicate": "(predicate IRI)",
  "type" : "(used to specify classification or regression)",
  "split_rate": [0.8,0.2,0.0]
}
```

Per le attività di previsione dei collegamenti RDF, gli oggetti di destinazione nell'array avranno l'aspetto seguente:

```
{
  "subject": "(source node type of an edge)",
  "predicate": "(relation type of an edge)",
  "object": "(destination node type of an edge)",
  "type" : "link_prediction",
  "split_rate": [0.8,0.2,0.0]
}
```

Gli oggetti di destinazione possono contenere i seguenti campi:

Indice

Campi in un oggetto di destinazione del grafo delle proprietà

Campo **node** (vertice) in un oggetto di destinazione

Etichetta del grafo delle proprietà di un nodo di destinazione (vertice). Un oggetto di destinazione deve contenere un elemento `node` o un elemento `edge`, ma non entrambi.

Un oggetto `node` accetta un singolo valore, come questo:

```
"node": "Movie"
```

Oppure, nel caso di un vertice con più etichette, può accettare un array di valori, come questo:

```
"node": ["Content", "Movie"]
```

Campo **`edge`** in un oggetto di destinazione del grafo delle proprietà

Specifica un arco di destinazione mediante le etichette del nodo iniziale, la propria etichetta e le etichette del nodo finale. Un oggetto di destinazione deve contenere un elemento `edge` o un elemento `node`, ma non entrambi.

Il valore di un campo `edge` è un array JSON di tre stringhe che rappresentano le etichette del grafo delle proprietà del nodo iniziale, l'etichetta del grafo delle proprietà dell'arco stesso e le etichette del grafo delle proprietà del nodo finale, in questo modo:

```
"edge": ["Person_A", "knows", "Person_B"]
```

Se il nodo iniziale e/o il nodo finale hanno più etichette, racchiudile in un array, in questo modo:

```
"edge": [ ["Admin", Person_A"], "knows", ["Admin", "Person_B"] ]
```

Campo **`property`** in un oggetto di destinazione del grafo delle proprietà

Specifica una proprietà del vertice o dell'arco di destinazione, in questo modo:

```
"property" : "rating"
```

Questo campo è obbligatorio, tranne quando l'attività di destinazione è la previsione dei collegamenti.

Campo **`type`** in un oggetto di destinazione del grafo delle proprietà

Indica il tipo di attività di destinazione da eseguire su `node` o `edge`, in questo modo:

```
"type" : "regression"
```

I tipi di attività supportati per i nodi sono:

- `classification`
- `regression`

I tipi di attività supportati per gli archi sono:

- `classification`
- `regression`
- `link_prediction`

Questo campo è obbligatorio.

Campo **`split_rate`** in un oggetto di destinazione del grafo delle proprietà

(Facoltativo): una stima delle proporzioni di nodi o archi che verranno utilizzate rispettivamente nelle fasi di addestramento, convalida e test. Queste proporzioni sono rappresentate da un array JSON di tre numeri compresi tra zero e uno che si sommano a uno:

```
"split_rate": [0.7, 0.1, 0.2]
```

Se non si specifica il campo `split_rate` facoltativo, il valore stimato predefinito è `[0.9, 0.1, 0.0]`.

Campo **`separator`** in un oggetto di destinazione del grafo delle proprietà

(Facoltativo): utilizzato con un'attività di classificazione.

Il campo `separator` specifica un carattere utilizzato per suddividere il valore di una proprietà di destinazione in più valori categoriali quando viene utilizzato per archiviare più valori di categoria in una stringa. Ad esempio:

```
"separator": "|"
```

La presenza di un campo `separator` indica che l'attività è un'attività di classificazione con più destinazioni.

Campi in un oggetto di destinazione RDF

Campo **`node`** in un oggetto di destinazione RDF

Definisce il tipo di nodo dei nodi di destinazione. Utilizzato con attività di classificazione dei nodi o attività di regressione dei nodi. Il tipo di nodo di un nodo in RDF è definito da:

```
  node_id, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, node_type
```

Un oggetto node RDF accetta solo un singolo valore, come questo:

```
  "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

Campo **`subject`** in un oggetto di destinazione RDF

Per le attività di previsione dei collegamenti, `subject` definisce il tipo di nodo di origine degli archi di destinazione.

```
  "subject": "http://aws.amazon.com/neptune/csv2rdf/class/Director"
```

> **Note**
>
> Per le attività di previsione dei collegamenti, `subject` deve essere utilizzato insieme a `predicate` e `object`. Se uno di questi tre elementi non viene specificato, tutti gli archi vengono considerati come destinazione dell'addestramento.

Campo **`predicate`** in un oggetto di destinazione RDF

Per le attività di classificazione e regressione dei nodi, `predicate` definisce quali dati letterali vengono utilizzati come funzionalità del nodo di destinazione di un nodo di destinazione.

```
  "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/genre"
```

> **Note**
>
> Se i nodi di destinazione hanno un solo predicato che definisce la funzionalità del nodo di destinazione, il campo `predicate` può essere omesso.

Per le attività di previsione dei collegamenti, `predicate` definisce il tipo di nodo di relazione degli archi di destinazione.

```
"predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/direct"
```

> **Note**
>
> Per le attività di previsione dei collegamenti, `predicate` deve essere utilizzato insieme a `subject` e `object`. Se uno di questi tre elementi non viene specificato, tutti gli archi vengono considerati come destinazione dell'addestramento.

Campo **`object`** in un oggetto di destinazione RDF

Per le attività di previsione dei collegamenti, `object` definisce il tipo di nodo di destinazione degli archi di destinazione:

```
"object": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

> **Note**
>
> Per le attività di previsione dei collegamenti, `object` deve essere utilizzato insieme a `subject` e `predicate`. Se uno di questi tre elementi non viene specificato, tutti gli archi vengono considerati come destinazione dell'addestramento.

Campo **`type`** in un oggetto di destinazione RDF

Indica il tipo di attività di destinazione da eseguire, in questo modo:

```
"type" : "regression"
```

I tipi di attività supportati per i dati RDF sono:

- `link_prediction`
- `classification`
- `regression`

Questo campo è obbligatorio.

Campo **`split_rate`** in un oggetto di destinazione del grafo delle proprietà

(Facoltativo): una stima delle proporzioni di nodi o archi che verranno utilizzate rispettivamente nelle fasi di addestramento, convalida e test. Queste proporzioni sono rappresentate da un array JSON di tre numeri compresi tra zero e uno che si sommano a uno:

```
"split_rate": [0.7, 0.1, 0.2]
```

Se non si specifica il campo `split_rate` facoltativo, il valore stimato predefinito è `[0.9, 0.1, 0.0]`.

Campo `features` in `neptune_ml`

I valori delle proprietà e i valori letterali RDF sono disponibili in formati e tipi di dati diversi. Per ottenere prestazioni ottimali nel machine learning, è essenziale convertire tali valori in codifiche numeriche note come funzionalità.

Neptune ML esegue l'estrazione e la codifica delle funzionalità come parte dei passaggi di esportazione ed elaborazione dei dati, come descritto in Codifica delle funzionalità in Neptune ML.

Per i set di dati con grafi di proprietà, il processo di esportazione deduce automaticamente le funzionalità `auto` delle proprietà stringa e delle proprietà numeriche contenenti più valori. Per le proprietà numeriche contenenti valori singoli, deduce funzionalità `numerical`. Per le proprietà data deduce funzionalità `datetime`.

Se desideri sovrascrivere una specifica di funzionalità dedotta automaticamente o aggiungere una specifica numerica del bucket, TF-IDF, fastText o SBERT per una proprietà, puoi controllare la codifica della funzionalità utilizzando il campo features.

> **Note**
>
> È possibile utilizzare il campo `features` solo per controllare le specifiche delle funzionalità per i dati dei grafi di proprietà, non per i dati RDF.

Per il testo in formato libero, Neptune ML può utilizzare diversi modelli per convertire la sequenza di token in un valore della proprietà stringa in un vettore di valori reali a dimensione fissa:

- `text_fasttext`: usa la codifica fastText. Si tratta della codifica consigliata per le funzionalità che utilizzano una e solo una delle cinque lingue supportate da fastText.
- `text_sbert`: usa i modelli di codifica Sentence BERT (SBERT). Si tratta della codifica consigliata per il testo non supportato da `text_fasttext`.
- `text_word2vec`: usa gli algoritmi Word2Vec originariamente pubblicati da Google per codificare il testo. Word2Vec supporta solo la lingua inglese.
- `text_tfidf`: usa un vettorizzatore TF-IDF (Term Frequency—Inverse Document Frequency) per la codifica del testo. La codifica TF-IDF supporta funzionalità statistiche non supportate da altre codifiche.

Il campo `features` contiene un array JSON di funzionalità delle proprietà dei nodi. Gli oggetti nell'array possono contenere i seguenti campi:

Indice

Campo **node** in **features**

Il campo node specifica l'etichetta del grafo delle proprietà di un vertice della funzionalità. Ad esempio:

```
  "node": "Person"
```

Se un vertice ha più etichette, racchiudile in un array. Ad esempio:

```
  "node": ["Admin", "Person"]
```

Campo **edge** in **features**

Il campo edge specifica il tipo di arco di un arco della funzionalità. Un tipo di arco è costituito da un array contenente le etichette del grafo delle proprietà del vertice di origine, l'etichetta del grafo delle proprietà dell'arco e le etichette del grafo delle proprietà del vertice di destinazione. È necessario fornire tutti e tre i valori quando si specifica una funzionalità dell'arco. Ad esempio:

```
"edge": ["User", "reviewed", "Movie"]
```

Se un vertice di origine o di destinazione di un tipo di arco ha più etichette, racchiudile in un altro array. Ad esempio:

```
"edge": [["Admin", "Person"]. "edited", "Post"]
```

Campo **property** in **features**

Usa il parametro property per specificare una proprietà del vertice identificato dal parametro node. Ad esempio:

```
"property" : "age"
```

Valori possibili del campo **type** per le funzionalità

Il parametro type specifica il tipo di funzionalità da definire. Ad esempio:

```
"type": "bucket_numerical"
```

Valori possibili del parametro **type**

- **"auto"**: specifica che Neptune ML deve rilevare automaticamente il tipo di proprietà e applicare una codifica delle funzionalità appropriata. Una funzionalità auto può anche avere un campo separator facoltativo.

  Per informazioni, consultare Codifica delle funzionalità auto in Neptune ML.
- **"category"**: questa codifica della funzionalità rappresenta il valore di una proprietà come una di varie categorie. In altre parole, la funzionalità può accettare uno o più valori discreti. Una funzionalità category può anche avere un campo separator facoltativo.

  Per informazioni, consultare Funzionalità categoriali in Neptune ML.

- **`"numerical"`**: questa codifica di funzionalità rappresenta i valori delle proprietà numeriche come numeri in un intervallo continuo in cui "maggiore di" e "minore di" hanno un significato.

  Una funzionalità `numerical` può anche avere campi `norm`, `imputer` e `separator`.

  Per informazioni, consultare Funzionalità numeriche in Neptune ML.

- **`"bucket_numerical"`**: questa codifica della funzionalità divide i valori delle proprietà numeriche in un set di bucket o categorie.

  Ad esempio, puoi codificare l'età delle persone in 4 bucket: bambini (0-20), giovani adulti (20-40), persone di mezza età (40-60 anni) e anziani (dai 60 anni in su).

  Una funzionalità `bucket_numerical` richiede un elemento `range` e un campo `bucket_cnt` e può facoltativamente includere anche un campo `imputer` e/o `slide_window_size`.

  Per informazioni, consultare Funzionalità numeriche per bucket in Neptune ML.

- **`"datetime"`**: questa codifica di funzionalità rappresenta il valore di una proprietà datetime come array di queste funzionalità categoriali: anno, mese, giorno della settimana e ora.

  Una o più di queste quattro categorie possono essere eliminate utilizzando il parametro `datetime_parts`.

  Per informazioni, consultare Funzionalità datetime in Neptune ML.

- **`"text_fasttext"`**: questa codifica di funzionalità converte i valori delle proprietà costituiti da frasi o testo in formato libero in vettori numerici usando modelli fastText. Supporta cinque lingue, ovvero inglese (`en`), cinese (`zh`), hindi (`hi`), spagnolo (`es`) e francese (`fr`). Per i valori delle proprietà di testo in ognuna di queste cinque lingue, `text_fasttext` è la codifica consigliata. Tuttavia, non è in grado di gestire i casi in cui la stessa frase contiene parole in più di una lingua.

  Per lingue diverse da quelle supportate da fastText, usa la codifica `text_sbert`.

  Se sono presenti molte stringhe di testo con valori delle proprietà più lunghe, ad esempio, di 120 token, usa il campo `max_length` per limitare il numero di token in ogni stringa codificata tramite `"text_fasttext"`.

  Per informazioni, consultare Codifica fastText dei valori delle proprietà di testo in Neptune ML.

- **`"text_sbert"`**: questa codifica converte i valori delle proprietà del testo in vettori numerici utilizzando i modelli Sentence BERT (SBERT). Neptune supporta due metodi SBERT, ovvero `text_sbert128`, che è il metodo predefinito se si specifica solo `text_sbert` e

`text_sbert512`. La differenza tra i due sta nel numero massimo di token che vengono codificati in una proprietà del testo. La codifica `text_sbert128` codifica solo i primi 128 token, mentre `text_sbert512` codifica fino a 512 token. Di conseguenza, l'utilizzo di `text_sbert512` può richiedere tempi di elaborazione superiori rispetto a `text_sbert128`. Entrambi i metodi sono più lenti di `text_fasttext`.

I metodi `text_sbert*` supportano molte lingue e possono codificare una frase che contiene più di una lingua.

Per informazioni, consultare Codifica Sentence BERT (SBERT) delle funzionalità di testo in Neptune ML.

- **`"text_word2vec"`**: questa codifica converte i valori delle proprietà del testo in vettori numerici utilizzando gli algoritmi Word2Vec. Supporta solo la lingua inglese.

  Per informazioni, consultare Codifica Word2Vec delle funzionalità di testo in Neptune ML.

- **`"text_tfidf"`**: questa codifica converte i valori delle proprietà del testo in vettori numerici utilizzando un vettorizzatore TF-IDF (Term Frequency—Inverse Document Frequency).

  È possibile definire i parametri della codifica di una funzionalità `text_tfidf` utilizzando il campo `ngram_range`, il campo `min_df` e il campo `max_features`.

  Per informazioni, consultare Codifica TF-IDF delle funzionalità di testo in Neptune ML.

- **`"none"`**: usando il tipo `none` non avviene alcuna codifica delle funzionalità. I valori della proprietà non elaborati vengono invece analizzati e salvati.

  Usa `none` solo se intendi eseguire una codifica personalizzata delle funzionalità come parte dell'addestramento personalizzato del modello.

Campo **`norm`**

Questo campo è obbligatorio per le funzioni numeriche. Specifica un metodo di normalizzazione da utilizzare sui valori numerici:

```
"norm": "min-max"
```

Sono supportati i seguenti metodi di normalizzazione:

- "min-max": normalizza ogni valore sottraendo da esso il valore minimo e poi dividendolo per la differenza tra il valore massimo e il valore minimo.
- "standard": normalizza ogni valore dividendolo per la somma di tutti i valori.
- "none": non normalizza i valori numerici durante la codifica.

Per informazioni, consultare Funzionalità numeriche in Neptune ML.

Campo **`language`**

Il campo della lingua specifica la lingua utilizzata nei valori delle proprietà del testo. Il suo utilizzo dipende dal metodo di codifica del testo:

- Per la codifica `text_fasttext` questo campo è obbligatorio e deve specificare una delle seguenti lingue:
  - `en` (inglese)
  - `zh` (cinese)
  - `hi` (hindi)
  - `es` (spagnolo)
  - `fr` (francese)
- Per la codifica `text_sbert` questo campo non viene utilizzato, poiché la codifica SBERT è multilingue.
- Per la codifica `text_word2vec` questo campo è facoltativo, poiché `text_word2vec` supporta solo la lingua inglese. Se presente, deve specificare il nome del modello linguistico inglese:

  ```
  "language" : "en_core_web_lg"
  ```

- Per la codifica `text_tfidf` questo campo non viene utilizzato.

Campo **`max_length`**

Il campo `max_length` è facoltativo per le funzionalità `text_fasttext`, in quanto specifica il numero massimo di token in una funzionalità di testo di input che verranno codificati. Il testo di input più lungo di `max_length` viene troncato. Ad esempio, l'impostazione di max_length su 128 indica che tutti i token successivi al 128° in una sequenza di testo verranno ignorati:

```
"max_length": 128
```

Campo **`separator`**

Questo campo viene utilizzato facoltativamente con le funzionalità `category`, `numerical` e `auto`. Specifica un carattere che può essere utilizzato per suddividere il valore di una proprietà in più valori categoriali o valori numerici:

```
"separator": ";"
```

Utilizza il campo `separator` solo quando la proprietà archivia più valori delimitati in una singola stringa, ad esempio `"Actor;Director"` o `"0.1;0.2"`.

Consulta le sezioni Funzionalità categoriali, Funzionalità numeriche e Codifica auto.

Campo **`range`**

Questo campo è obbligatorio per le funzionalità `bucket_numerical`. Specifica l'intervallo di valori numerici che devono essere suddivisi in bucket, nel formato: `[`*`lower-bound`*`, `*`upper-bound`*`]`.

```
"range" : [20, 100]
```

Se il valore di una proprietà è minore del limite inferiore, viene assegnato al primo bucket o, se è maggiore del limite superiore, viene assegnato all'ultimo bucket.

Per informazioni, consultare Funzionalità numeriche per bucket in Neptune ML.

Campo **`bucket_cnt`**

Questo campo è obbligatorio per le funzionalità `bucket_numerical`. Specifica il numero di bucket in cui deve essere suddiviso l'intervallo numerico definito dal parametro `range`:

```
"bucket_cnt": 10
```

Per informazioni, consultare Funzionalità numeriche per bucket in Neptune ML.

Campo **`slide_window_size`**

Questo campo viene usato facoltativamente con le funzionalità `bucket_numerical` per assegnare valori a più di un bucket:

```
"slide_window_size": 5
```

Per le finestre scorrevoli, Neptune ML prende le dimensioni della finestra **s** e trasforma ogni valore **v** numerico di una proprietà in un intervallo da `v - s/2` a `v + s/2` . Il valore viene quindi assegnato a ogni bucket a cui si sovrappone l'intervallo.

Per informazioni, consultare Funzionalità numeriche per bucket in Neptune ML.

Campo **`imputer`**

Questo campo viene utilizzato facoltativamente con le funzionalità `numerical` e `bucket_numerical` per fornire una tecnica di imputazione per inserire i valori mancanti:

```
"imputer": "mean"
```

Le tecniche di imputazione supportate sono:

- `"mean"`
- `"median"`
- `"most-frequent"`

Se non si include il parametro imputer, la pre-elaborazione dei dati si interrompe e termina quando viene rilevato un valore mancante.

Consultare Funzionalità numeriche in Neptune ML e Funzionalità numeriche per bucket in Neptune ML.

Campo **`max_features`**

Questo campo viene utilizzato facoltativamente dalle funzionalità `text_tfidf` per specificare il numero massimo di termini da codificare:

```
"max_features": 100
```

Se viene ad esempio impostato su 100, il vettorizzatore TF-IDF codificherà solo i 100 termini più comuni. Il valore predefinito, se si include `max_features`, è 5.000.

Per informazioni, consultare Codifica TF-IDF delle funzionalità di testo in Neptune ML.

Campo **`min_df`**

Questo campo viene utilizzato facoltativamente dalle funzionalità `text_tfidf` per specificare la frequenza minima nel documento dei termini da codificare:

```
"min_df": 5
```

Se viene ad esempio impostato su 5, indica che un termine deve comparire in almeno 5 valori di proprietà diversi per poter essere codificato.

Il valore predefinito, se non si include il parametro `min_df` è 2.

Per informazioni, consultare Codifica TF-IDF delle funzionalità di testo in Neptune ML.

Campo **`ngram_range`**

Questo campo viene utilizzato facoltativamente dalle funzionalità `text_tfidf` per specificare le dimensioni delle sequenze di parole o token da considerare come potenziali termini individuali da codificare:

```
"ngram_range": [2, 4]
```

Il valore `[2, 4]` specifica che le sequenze di 2, 3 e 4 parole devono essere considerate come potenziali termini individuali.

L'impostazione predefinita, se non si imposta `ngram_range` in modo esplicito è `[1, 1]`, ovvero solo singole parole o token vengono considerati come termini da codificare.

Per informazioni, consultare Codifica TF-IDF delle funzionalità di testo in Neptune ML.

Campo **`datetime_parts`**

Questo campo viene utilizzato facoltativamente dalle funzionalità `datetime` per specificare quali parti del valore datetime codificare in modo categoriale:

```
"datetime_parts": ["weekday", "hour"]
```

Se non si include `datetime_parts`, per impostazione predefinita Neptune ML codifica le parti relative all'anno, al mese, al giorno della settimana e all'ora del valore datetime. Il valore

`["weekday", "hour"]` indica che solo i valori di data/ora del giorno della settimana e dell'ora devono essere codificati in modo categoriale nella funzionalità.

Se una delle parti non ha più di un valore univoco nel set di addestramento, non viene codificata.

Per informazioni, consultare Funzionalità datetime in Neptune ML.

Esempi di utilizzo dei parametri in **`additionalParams`** per l'ottimizzazione della configurazione di addestramento del modello

Indice

- Definizione di un'attività di classificazione dei nodi per la configurazione dell'addestramento dei modelli
- Definizione di un'attività di regressione dei nodi per la configurazione dell'addestramento dei modelli
- Definizione di un'attività di previsione dei collegamenti per archi particolari
- Definizione di un'attività di previsione dei collegamenti per tutti gli archi

Esempi di grafi di proprietà con **`additionalParams`**

Definizione di un coefficiente di divisione per la configurazione dell'addestramento dei modelli

Nell'esempio seguente, il parametro `split_rate` imposta il coefficiente di divisione predefinito per l'addestramento dei modelli. Se non viene specificato alcun coefficiente di divisione predefinito, l'addestramento utilizza un valore di [0,9, 0,1, 0,0]. È possibile sovrascrivere il valore predefinito per ogni destinazione specificando un valore `split_rate` per ogni destinazione.

Nell'esempio seguente il campo `default split_rate` indica che occorre usare un coefficiente di divisione di `[0.7,0.1,0.2]`, a meno che non venga sovrascritto per ogni singola destinazione:"

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
    "targets": [
      (...)
    ],
    "features": [
      (...)
    ]
  }
}
```

Definizione di un'attività di classificazione dei nodi per la configurazione dell'addestramento dei modelli

Per indicare qual è la proprietà del nodo contenente esempi etichettati ai fini dell'addestramento, aggiungi un elemento di classificazione dei nodi all'array `targets`, usando `"type" : "classification"`. Aggiungi un campo `split_rate` per sovrascrivere il coefficiente di divisione predefinito.

Nell'esempio seguente la destinazione node indica che la proprietà genre di ogni nodo Movie deve essere trattata come un'etichetta della classe nodo. Il valore split_rate sovrascrive il coefficiente di divisione predefinito:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "genre",
        "type": "classification",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Definizione di un'attività di classificazione dei nodi multi-classe per la configurazione dell'addestramento dei modelli

Per indicare qual è la proprietà del nodo contenente più esempi etichettati ai fini dell'addestramento, aggiungi un elemento di classificazione dei nodi all'array targets usando "type" : "classification" e separator per specificare un carattere da usare per dividere un valore della proprietà di destinazione in più valori categoriali. Aggiungi un campo split_rate per sovrascrivere il coefficiente di divisione predefinito.

Nell'esempio seguente la destinazione node indica che la proprietà genre di ogni nodo Movie deve essere trattata come un'etichetta della classe nodo. Il campo separator indica che ogni proprietà di genere contiene più valori separati da punto e virgola:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "genre",
        "type": "classification",
```

```
        "separator": ";"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Definizione di un'attività di regressione dei nodi per la configurazione dell'addestramento dei modelli

Per indicare qual è la proprietà del nodo contenente le regressioni etichettate ai fini dell'addestramento, aggiungi un elemento di regressione dei nodi all'array targets usando `"type" : "regression"`. Aggiungi un campo split_rate per sovrascrivere il coefficiente di divisione predefinito.

La destinazione node seguente indica che la proprietà `rating` di ogni nodo `Movie` deve essere trattata come un'etichetta di regressione del nodo.

```
    "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "rating",
        "type" : "regression",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
    "features": [
      ...
    ]
  }
}
```

Definizione di un'attività di classificazione degli archi per la configurazione dell'addestramento dei modelli

Per indicare qual è la proprietà dell'arco contenente esempi etichettati ai fini dell'addestramento, aggiungi un elemento arco all'array `targets`, usando `"type" : "regression"`. Aggiungi un campo split_rate per sovrascrivere il coefficiente di divisione predefinito.

La destinazione edge seguente indica che la proprietà `metAtLocation` di ogni arco `knows` deve essere trattata come un'etichetta della classe arco:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "knows", "Person"],
        "property": "metAtLocation",
        "type": "classification"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Definizione di un'attività di classificazione degli archi multi-classe per la configurazione dell'addestramento dei modelli

Per indicare qual è la proprietà dell'arco contenente più esempi etichettati ai fini dell'addestramento, aggiungi un elemento edge all'array `targets` usando `"type" : "classification"` e un campo `separator` per specificare un carattere da usare per dividere un valore della proprietà di destinazione in più valori categoriali. Aggiungi un campo `split_rate` per sovrascrivere il coefficiente di divisione predefinito.

La destinazione edge seguente indica che la proprietà `sentiment` di ogni arco `repliedTo` deve essere trattata come un'etichetta della classe arco. Il campo separatore indica che ogni proprietà sentiment contiene più valori separati da virgole:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "repliedTo", "Message"],
        "property": "sentiment",
        "type": "classification",
        "separator": ","
```

```
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Definizione di un'attività di regressione degli archi per la configurazione dell'addestramento dei modelli

Per indicare qual è la proprietà dell'arco contenente esempi di regressione etichettati ai fini dell'addestramento, aggiungi un elemento `edge` all'array `targets`, usando `"type" : "regression"`. Aggiungi un campo `split_rate` per sovrascrivere il coefficiente di divisione predefinito.

La destinazione `edge` seguente indica che la proprietà `rating` di ogni arco `reviewed` deve essere trattata come regressione degli archi:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "reviewed", "Movie"],
        "property": "rating",
        "type" : "regression"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Definizione di un'attività di previsione dei collegamenti per la configurazione dell'addestramento dei modelli

Per indicare gli archi da usare ai fini dell'addestramento delle previsioni dei collegamenti, aggiungi un elemento edge all'array targets usando `"type" : "link_prediction"`. Aggiungi un campo `split_rate` per sovrascrivere il coefficiente di divisione predefinito.

La destinazione edge seguente indica che è necessario usare gli archi `cites` per la previsione dei collegamenti:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Article", "cites", "Article"],
        "type" : "link_prediction"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Definizione di una funzionalità bucket numerico

È possibile specificare una funzionalità dati numerici per una proprietà del nodo aggiungendo `"type": "bucket_numerical"` all'array `features`.

La funzionalità node seguente indica che la proprietà `age` di ogni nodo `Person` deve essere trattata come una funzionalità bucket numerico:

```
  "additionalParams": {
  "neptune_ml": {
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Person",
        "property": "age",
        "type": "bucket_numerical",
        "range": [1, 100],
        "bucket_cnt": 5,
        "slide_window_size": 3,
        "imputer": "median"
      }
    ]
```

```
  }
}
```

Definizione di una funzionalità **`Word2Vec`**

È possibile specificare una funzionalità `Word2Vec` per una proprietà del nodo aggiungendo `"type": "text_word2vec"` all'array `features`.

La funzionalità node seguente indica che la proprietà `description` di ogni nodo `Movie` deve essere trattata come una funzionalità `Word2Vec`:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_word2vec",
        "language": "en_core_web_lg"
      }
    ]
  }
}
```

Definizione di una funzionalità **`FastText`**

È possibile specificare una funzionalità `FastText` per una proprietà del nodo aggiungendo `"type": "text_fasttext"` all'array `features`. Il campo `language` è obbligatorio e deve specificare uno dei codici di lingua seguenti:

- `en` (inglese)
- `zh` (cinese)
- `hi` (hindi)
- `es` (spagnolo)
- `fr` (francese)

Tieni presente che la codifica `text_fasttext` non può gestire più di una lingua alla volta in una funzionalità.

La funzionalità node seguente indica che la proprietà `description` francese di ogni nodo `Movie` deve essere trattata come una funzionalità `FastText`:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_fasttext",
        "language": "fr",
        "max_length": 1024
      }
    ]
  }
}
```

Definizione di una funzionalità **`Sentence BERT`**

È possibile specificare una funzionalità `Sentence BERT` per una proprietà del nodo aggiungendo `"type": "text_sbert"` all'array `features`. Non è necessario specificare la lingua, poiché il metodo codifica automaticamente le funzionalità di testo utilizzando un modello linguistico multilingue.

La funzionalità node seguente indica che la proprietà `description` di ogni nodo `Movie` deve essere trattata come una funzionalità `Sentence BERT`:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
```

```
        "property": "description",
        "type": "text_sbert128",
      }
    ]
  }
}
```

Definizione di una funzionalità **`TF-IDF`**

È possibile specificare una funzionalità `TF-IDF` per una proprietà del nodo aggiungendo `"type": "text_tfidf"` all'array `features`.

La funzionalità `node` seguente indica che la proprietà `bio` di ogni nodo `Person` deve essere trattata come una funzionalità `TF-IDF`:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "bio",
        "type": "text_tfidf",
        "ngram_range": [1, 2],
        "min_df": 5,
        "max_features": 1000
      }
    ]
  }
}
```

Definizione di una funzionalità **`datetime`**

Il processo di esportazione deduce automaticamente le funzionalità `datetime` per le proprietà data. Tuttavia, per limitare l'uso di `datetime_parts` per una funzionalità `datetime` o ignorare una specifica di funzionalità affinché una proprietà che normalmente verrebbe trattata come una funzionalità `auto` venga trattata esplicitamente come una funzionalità `datetime`, puoi aggiungere l'elemento `"type": "datetime"` all'array features.

La funzionalità node seguente indica che la proprietà `createdAt` di ogni nodo `Post` deve essere trattata come una funzionalità `datetime`:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Post",
        "property": "createdAt",
        "type": "datetime",
        "datetime_parts": ["month", "weekday", "hour"]
      }
    ]
  }
}
```

Definizione di una funzionalità **`category`**

Il processo di esportazione deduce automaticamente le funzionalità `auto` delle proprietà stringa e delle proprietà numeriche contenenti più valori. Per le proprietà numeriche contenenti valori singoli, deduce funzionalità `numerical`. Per le proprietà data deduce funzionalità `datetime`.

Se desideri sovrascrivere la specifica di una funzionalità affinché una proprietà venga trattata come una funzionalità categoriale, aggiungi un elemento `"type": "category"` all'array features. Se la proprietà contiene più valori, includi un campo `separator`. Ad esempio:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Post",
        "property": "tag",
        "type": "category",
        "separator": "|"
```

```
      }
    ]
  }
}
```

Definizione di una funzionalità **`numerical`**

Il processo di esportazione deduce automaticamente le funzionalità `auto` delle proprietà stringa e delle proprietà numeriche contenenti più valori. Per le proprietà numeriche contenenti valori singoli, deduce funzionalità `numerical`. Per le proprietà data deduce funzionalità `datetime`.

Se desideri sovrascrivere la specifica di una funzionalità affinché una proprietà venga trattata come una funzionalità `numerical`, aggiungi `"type": "numerical"` all'array features. Se la proprietà contiene più valori, includi un campo `separator`. Ad esempio:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Recording",
        "property": "duration",
        "type": "numerical",
        "separator": ","
      }
    ]
  }
}
```

Definizione di una funzionalità **`auto`**

Il processo di esportazione deduce automaticamente le funzionalità `auto` delle proprietà stringa e delle proprietà numeriche contenenti più valori. Per le proprietà numeriche contenenti valori singoli, deduce funzionalità `numerical`. Per le proprietà data deduce funzionalità `datetime`.

Se desideri sovrascrivere la specifica di una funzionalità affinché una proprietà venga trattata come una funzionalità `auto`, aggiungi `"type": "auto"` all'array features. Se la proprietà contiene più valori, includi un campo `separator`. Ad esempio:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "User",
        "property": "role",
        "type": "auto",
        "separator": ","
      }
    ]
  }
}
```

Esempi di grafi RDF con **`additionalParams`**

Definizione di un coefficiente di divisione per la configurazione dell'addestramento dei modelli

Nell'esempio seguente, il parametro `split_rate` imposta il coefficiente di divisione predefinito per l'addestramento dei modelli. Se non viene specificato alcun coefficiente di divisione predefinito, l'addestramento utilizza un valore di [0,9, 0,1, 0,0]. È possibile sovrascrivere il valore predefinito per ogni destinazione specificando un valore `split_rate` per ogni destinazione.

Nell'esempio seguente il campo `default split_rate` indica che occorre usare un coefficiente di divisione di `[0.7,0.1,0.2]`, a meno che non venga sovrascritto per ogni singola destinazione:"

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
    "targets": [
      (...)
    ]
  }
}
```

Definizione di un'attività di classificazione dei nodi per la configurazione dell'addestramento dei modelli

Per indicare qual è la proprietà del nodo contenente esempi etichettati ai fini dell'addestramento, aggiungi un elemento di classificazione dei nodi all'array `targets`, usando `"type" : "classification"`. Aggiungi un campo node per indicare il tipo di nodo dei nodi di destinazione. Aggiungi un campo `predicate` per definire i dati letterali da usare come funzionalità del nodo di destinazione del nodo di destinazione. Aggiungi un campo `split_rate` per sovrascrivere il coefficiente di divisione predefinito.

Nell'esempio seguente la destinazione `node` indica che la proprietà `genre` di ogni nodo `Movie` deve essere trattata come un'etichetta della classe nodo. Il valore `split_rate` sovrascrive il coefficiente di divisione predefinito:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/genre",
        "type": "classification",
        "split_rate": [0.7,0.1,0.2]
      }
    ]
  }
}
```

Definizione di un'attività di regressione dei nodi per la configurazione dell'addestramento dei modelli

Per indicare qual è la proprietà del nodo contenente le regressioni etichettate ai fini dell'addestramento, aggiungi un elemento di regressione dei nodi all'array targets usando `"type" : "regression"`. Aggiungi un campo `node` per indicare il tipo di nodo dei nodi di destinazione. Aggiungi un campo `predicate` per definire i dati letterali da usare come funzionalità del nodo di destinazione del nodo di destinazione. Aggiungi un campo `split_rate` per sovrascrivere il coefficiente di divisione predefinito.

La destinazione `node` seguente indica che la proprietà `rating` di ogni nodo `Movie` deve essere trattata come un'etichetta di regressione del nodo.

```
    "additionalParams": {
```

```
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/rating",
        "type": "regression",
        "split_rate": [0.7,0.1,0.2]
      }
    ]
  }
}
```

Definizione di un'attività di previsione dei collegamenti per archi particolari

Per indicare gli archi da usare ai fini dell'addestramento delle previsioni dei collegamenti, aggiungi un elemento edge all'array targets usando `"type" : "link_prediction"`. Aggiungi i campi `subject`, `predicate` e `object` per specificare il tipo di arco. Aggiungi un campo `split_rate` per sovrascrivere il coefficiente di divisione predefinito.

La destinazione `edge` seguente indica che è necessario usare gli archi `directed` che connettono `Directors` a `Movies` per la previsione dei collegamenti:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "subject": "http://aws.amazon.com/neptune/csv2rdf/class/Director",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/directed",
        "object": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "type" : "link_prediction"
      }
    ]
  }
}
```

Definizione di un'attività di previsione dei collegamenti per tutti gli archi

Per indicare che è necessario usare tutti gli archi ai fini dell'addestramento delle previsioni dei collegamenti, aggiungi un elemento `edge` all'array targets usando `"type" :`

`"link_prediction"`. Non aggiungere i campi `subject`, `predicate` o `object`. Aggiungi un campo `split_rate` per sovrascrivere il coefficiente di divisione predefinito.

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "type" : "link_prediction"
      }
    ]
  }
}
```

Elaborazione dei dati del grafo esportati da Neptune per l'addestramento

La fase di elaborazione dei dati utilizza i dati del grafo di Neptune creati dal processo di esportazione e crea le informazioni utilizzate da Deep Graph Library (DGL) durante l'addestramento. Include l'esecuzione di varie mappature e trasformazioni dei dati:

- Analisi di nodi e archi per creare i file di mappatura di grafo e ID richiesti da DGL.
- Conversione delle proprietà dei nodi e degli archi nelle funzionalità di nodi e archi richieste da DGL.
- Divisione dei dati in set di addestramento, convalida e test.

Gestione della fase di elaborazione dei dati per Neptune ML

Dopo aver esportato da Neptune i dati da utilizzare per l'addestramento dei modelli, puoi avviare un processo di elaborazione dati utilizzando un comando `curl` (o `awscurl`) come il seguente:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)",
        "configFileName" : "training-job-configuration.json"
      }'
```

I dettagli su come utilizzare questo comando sono illustrati in Comando dataprocessing, insieme a informazioni su come recuperare lo stato di un processo in esecuzione, come arrestare un processo in esecuzione e come elencare tutti i processi in esecuzione.

Elaborazione dei dati del grafo aggiornati per Neptune ML

È inoltre possibile fornire un `previousDataProcessingJobId` all'API per garantire che il nuovo processo di elaborazione dei dati utilizzi lo stesso metodo di elaborazione di un processo precedente. Questo è necessario quando si desidera ottenere previsioni per i dati del grafo aggiornati in Neptune,

riaddestrando il modello precedente sui nuovi dati o ricalcolando gli artefatti del modello sui nuovi dati.

A tale scopo, puoi usare un comando `curl` (o `awscurl`) nel seguente modo:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{ "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
       "id" : "(a job ID for the new job)",
       "processedDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your output
 folder)",
       "previousDataProcessingJobId", "(the job ID of the previous data-processing
 job)"}'
```

Imposta il valore del parametro `previousDataProcessingJobId` sull'ID processo del processo di elaborazione dati precedente corrispondente al modello addestrato.

> **Note**
>
> Le eliminazioni dei nodi nel grafo aggiornato non sono attualmente supportate. Se i nodi sono stati rimossi in un grafo aggiornato, è necessario avviare un processo di elaborazione dei dati completamente nuovo anziché usare `previousDataProcessingJobId`.

Codifica delle funzionalità in Neptune ML

I valori delle proprietà sono disponibili in formati e tipi di dati diversi. Per ottenere prestazioni ottimali nel machine learning, è essenziale convertire tali valori in codifiche numeriche note come funzionalità.

Neptune ML esegue l'estrazione e la codifica delle funzionalità come parte dei passaggi di esportazione ed elaborazione dei dati, usando le tecniche di codifica delle funzionalità descritte qui.

> **Note**
>
> Se prevedi di implementare una codifica delle funzionalità personalizzata nell'implementazione di un modello personalizzato, puoi disabilitare la codifica automatica delle funzionalità nella fase di pre-elaborazione dei dati selezionando `none` come tipo di codifica delle funzionalità. Non verrà quindi eseguita alcuna codifica delle funzionalità sulla proprietà del nodo o dell'arco specifica e i valori delle proprietà non elaborati verranno analizzati e salvati in un dizionario. La pre-elaborazione dei dati crea comunque il grafo DGL dal set di dati esportato, ma il grafo DGL creato non dispone delle funzionalità pre-elaborate per l'addestramento.
>
> Usa questa opzione solo se intendi eseguire una codifica personalizzata delle funzionalità come parte dell'addestramento personalizzato del modello. Per informazioni dettagliate, consulta Modelli personalizzati in Neptune ML..

Funzionalità categoriali in Neptune ML

Una proprietà che può accettare uno o più valori distinti da un elenco fisso di valori possibili è una funzionalità categoriale. In Neptune ML le funzionalità categoriali vengono codificate mediante la codifica one-hot. L'esempio seguente mostra in che modo viene applicata la codifica one-hot al nome della proprietà di diversi alimenti in base alla relativa categoria:

```
 Food        Veg.   Meat   Fruit    Encoding
---------    ----   ----   -----    --------
 Apple         0      0      1         001
 Chicken       0      1      0         010
 Broccoli      1      0      0         100
```

> **Note**
>
> Il numero massimo di categorie in ogni funzionalità categoriale è 100. Se una proprietà ha più di 100 categorie di valore, solo le 99 più comuni vengono inserite in categorie distinte, mentre le altre vengono incluse in una categoria speciale denominata `OTHER`.

Funzionalità numeriche in Neptune ML

Qualsiasi proprietà i cui valori sono numeri reali può essere codificata come funzionalità numerica in Neptune ML. Le funzionalità numeriche sono codificate utilizzando numeri a virgola mobile.

È possibile specificare un metodo di normalizzazione dei dati da utilizzare per la codifica delle funzionalità numeriche, in questo modo: `"norm": "`*`normalization technique`*`"`. Sono supportate le seguenti tecniche di normalizzazione:

- "none": non normalizza i valori numerici durante la codifica.
- "min-max": normalizza ogni valore sottraendo da esso il valore minimo e poi dividendolo per la differenza tra il valore massimo e il valore minimo.
- "standard": normalizza ogni valore dividendolo per la somma di tutti i valori.

Funzionalità numeriche per bucket in Neptune ML

Invece di rappresentare una proprietà numeriche con numeri non elaborati, è possibile condensare i valori numerici in categorie. Ad esempio, è possibile suddividere l'età delle persone in categorie come bambini (0-20 anni), giovani adulti (20-40 anni), persone di mezza età (40-60 anni) e anziani (dai 60 anni in poi). Usando questi bucket numerici, si trasforma una proprietà numerica in una sorta di funzionalità categoriale.

In Neptune ML per codificare una proprietà numerica come funzionalità numerica per bucket, devi specificare due elementi:

- Un intervallo numerico nel formato `"range": [`*`a`*`, `*`b`*`]` , dove a e b sono numeri interi.
- Un numero di bucket nel formato `"bucket_cnt": `*`c`* , dove c è il numero di bucket, anch'esso un numero intero.

Neptune ML calcola quindi le dimensioni di ogni bucket come `( b - a ) / c` e codifica ogni valore numerico come il numero dell'eventuale bucket in cui rientra. Qualsiasi valore inferiore a a viene considerato appartenente al primo bucket e qualsiasi valore superiore a b viene considerato appartenente all'ultimo bucket.

Facoltativamente, puoi anche far rientrare i valori numerici in più di un bucket, specificando le dimensioni di una finestra scorrevole, come questa: `"slide_window_size": s` , dove `s` è un numero. Neptune ML trasforma quindi ogni valore numerico `v` della proprietà in un intervallo da `v - s/2` a `v + s/2` e assegna il valore `v` a ogni bucket incluso nell'intervallo.

Infine, è facoltativamente possibile specificare un modo per inserire i valori mancanti per le funzionalità numeriche e le funzionalità numeriche per bucket. A tale scopo, si usa `"imputer": "imputation technique "`, dove la tecnica di imputazione è una tra `"mean"`, `"median"` o `"most-frequent"`. Se non si specifica un imputer, un valore mancante può causare l'interruzione dell'elaborazione.

Codifica delle funzionalità di testo in Neptune ML

Per il testo in formato libero, Neptune ML può utilizzare diversi modelli per convertire la sequenza di token in una stringa del valore della proprietà in un vettore di valori reali a dimensione fissa:

- `text_fasttext`: usa la codifica fastText. Si tratta della codifica consigliata per le funzionalità che utilizzano una e solo una delle cinque lingue supportate da fastText.
- `text_sbert`: usa i modelli di codifica Sentence BERT (SBERT). Si tratta della codifica consigliata per il testo non supportato da `text_fasttext`.
- `text_word2vec`: usa gli algoritmi Word2Vec originariamente pubblicati da Google per codificare il testo. Word2Vec supporta solo la lingua inglese.
- `text_tfidf`: usa un vettorizzatore TF-IDF (Term Frequency—Inverse Document Frequency) per la codifica del testo. La codifica TF-IDF supporta funzionalità statistiche non supportate da altre codifiche.

Codifica fastText dei valori delle proprietà di testo in Neptune ML

Neptune ML può utilizzare i modelli fastText per convertire i valori delle proprietà di testo in vettori di valori reali a dimensione fissa. Si tratta del metodo di codifica consigliato per i valori delle proprietà di testo in una delle cinque lingue supportate da fastText:

- `en` (inglese)

- `zh` (cinese)
- `hi` (hindi)
- `es` (spagnolo)
- `fr` (francese)

Tieni presente che fastText non può gestire frasi contenenti parole in più di una lingua.

Il metodo `text_fasttext` può facoltativamente usare un campo `max_length` che specifica il numero massimo di token nel valore di una proprietà di testo che verrà codificato, dopodiché la stringa viene troncata. In questo modo è possibile migliorare le prestazioni quando i valori delle proprietà di testo contengono stringhe lunghe, perché se non si specifica `max_length`, fastText codifica tutti i token indipendentemente dalla lunghezza della stringa.

Questo esempio specifica che i titoli dei film in francese sono codificati utilizzando fastText:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
        "feature": ["title", "title", "text_fasttext"],
        "language": "fr",
        "max_length": 1024
      }
    ]
  }
```

Codifica Sentence BERT (SBERT) delle funzionalità di testo in Neptune ML

Neptune ML può convertire la sequenza di token in un valore della proprietà stringa in un vettore di valori reali a dimensione fissa utilizzando i modelli Sentence BERT (SBERT). Neptune supporta due metodi SBERT: `text_sbert128`, che è il metodo predefinito se si specifica solo `text_sbert` e `text_sbert512`. La differenza tra i due sta nella lunghezza massima di una stringa di valori della proprietà di testo codificata. La codifica `text_sbert128` tronca le stringhe di testo dopo aver codificato 128 token, mentre `text_sbert512` tronca le stringhe di testo dopo aver codificato 512 token. Di conseguenza, l'utilizzo di `text_sbert512` richiede tempi di elaborazione superiori rispetto a `text_sbert128`. Entrambi i metodi sono più lenti di `text_fasttext`.

La codifica SBERT è multilingue, quindi non è necessario specificare una lingua per il testo del valore della proprietà da codificare. SBERT supporta molte lingue e può codificare una frase che contiene più di una lingua. Se si codificano valori di proprietà contenenti testo in una o più lingue non supportate da fastText, SBERT è il metodo di codifica consigliato.

L'esempio seguente specifica che i titoli dei film sono codificati come SBERT fino a un massimo di 128 token:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      { "feature": ["title", "title", "text_sbert128"] }
    ]
  }
```

Codifica Word2Vec delle funzionalità di testo in Neptune ML

Neptune ML può codificare i valori delle proprietà stringa come funzionalità Word2Vec (gli algoritmi Word2Vec sono stati inizialmente pubblicati da Google). Il metodo `text_word2vec` codifica i token in una stringa come vettore dense utilizzando uno dei modelli addestrati da spaCy. Supporta solo la lingua inglese utilizzando il modello en_core_web_lg.

L'esempio seguente specifica che i titoli dei film sono codificati con Word2Vec:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
        "feature": ["title", "title", "text_word2vec"],
        "language": "en_core_web_lg"
      }
    ]
  }
```

Nota che il campo della lingua è facoltativo, poiché il modello `en_core_web_lg` in lingua inglese è l'unico supportato da Neptune.

Codifica TF-IDF delle funzionalità di testo in Neptune ML

Neptune ML può codificare i valori delle proprietà di testo come funzionalità `text_tfidf`. Questa codifica converte la sequenza di parole nel testo in un vettore numerico utilizzando un vettorizzatore TF-IDF (Term Frequency—Inverse Document Frequency), seguito da un'operazione di riduzione della dimensionalità.

TF-IDF (Term Frequency—Inverse Document Frequency) è un valore numerico destinato a misurare l'importanza di una parola in un set di documenti. Viene calcolato dividendo il numero di volte in cui una parola compare in un determinato valore delle proprietà per il numero totale di tali valori delle proprietà in cui compare.

Ad esempio, se la parola "kiss" compare due volte nel titolo di un determinato film (ad esempio, "kiss kiss bang bang") e "kiss" compare nel titolo di 4 film in tutto, allora il valore TF-IDF di "kiss" nel titolo "kiss kiss bang bang" sarà `2 / 4`.

Il vettore creato inizialmente ha d dimensioni, dove d è il numero di termini univoci in tutti i valori delle proprietà di quel tipo. L'operazione di riduzione della dimensionalità utilizza una proiezione sparse casuale per ridurre tale numero a un massimo di 100. Viene quindi generato il vocabolario di un grafo unendo tutte le funzionalità `text_tfidf` al suo interno.

È possibile controllare il vettorizzatore TF-IDF in diversi modi:

- **`max_features`**: usando il parametro `max_features` è possibile limitare il numero di termini nelle funzionalità `text_tfidf` a quelli più comuni. Ad esempio, se si imposta `max_features` su 100, vengono inclusi solo i primi 100 termini più utilizzati. Il valore predefinito per `max_features`, se non viene impostato esplicitamente, è 5.000.
- **`min_df`**: usando il parametro `min_df` è possibile limitare il numero di termini nelle funzionalità `text_tfidf` a quelli con almeno una frequenza nel documento specificata. Se ad esempio si imposta `min_df` su 5, vengono utilizzati solo i termini che compaiono in almeno 5 valori delle proprietà diversi. Il valore predefinito per `min_df`, se non viene impostato esplicitamente, è 2.
- **`ngram_range`**: il parametro `ngram_range` determina quali combinazioni di parole vengono trattate come termini. Se ad esempio si imposta `ngram_range` su `[2, 4]`, nel titolo "kiss kiss bang bang" verranno trovati i 6 termini seguenti:
  - Termini di 2 parole: "kiss kiss", "kiss bang" e "bang bang".
  - Termini di 3 parole: "kiss kiss bang" e "kiss bang bang".
  - Termini di 4 parole: "kiss kiss bang bang".

L'impostazione predefinita per `ngram_range` è `[1, 1]`.

Funzionalità datetime in Neptune ML

Neptune ML può convertire parti dei valori della proprietà `datetime` in funzionalità categoriali codificandole come array one-hot. Usa il parametro `datetime_parts` per specificare una o più delle seguenti parti da codificare: `["year", "month", "weekday", "hour"]`. Se non imposti `datetime_parts`, per impostazione predefinita vengono codificate tutte e quattro le parti.

Ad esempio, se l'intervallo di valori datetime comprende gli anni dal 2010 al 2012, le quattro parti della voce datetime `2011-04-22 01:16:34` sono le seguenti:

- year: `[0, 1, 0]`.

  Poiché l'intervallo comprende solo 3 anni (2010, 2011 e 2012), l'array one-hot contiene tre voci, una per ogni anno.
- month: `[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]`.

  Qui, l'array one-hot ha una voce per ogni mese dell'anno.
- weekday: `[0, 0, 0, 0, 1, 0, 0]`.

  Lo standard ISO 8601 dichiara che il lunedì è il primo giorno della settimana e, poiché il 22 aprile 2011 era un venerdì, il valore one-hot del giorno della settimana nell'array si trova in quinta posizione.
- hour: `[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`.

  L'ora è impostata sulla 1:00 in un array one-hot a 24 membri.

Il giorno del mese, il minuto e il secondo non sono codificati in modo categoriale.

Se l'intervallo `datetime` totale in questione include solo date comprese in un singolo anno, non viene codificato alcun array `year`.

È possibile specificare una strategia di imputazione per inserire i valori `datetime` mancanti, utilizzando il parametro `imputer` e una delle strategie disponibili per le funzionalità numeriche.

Codifica delle funzionalità auto in Neptune ML

Anziché specificare manualmente i metodi di codifica delle funzionalità da utilizzare per le proprietà del grafo, puoi impostare `auto` come metodo di codifica delle funzionalità. Neptune ML tenta quindi di dedurre la migliore codifica delle funzionalità per ogni proprietà in base al tipo di dati sottostante.

Ecco alcune delle euristiche utilizzate da Neptune ML per selezionare le codifiche di funzionalità appropriate:

- Se la proprietà ha solo valori numerici e può essere convertita in tipi di dati numerici, Neptune ML generalmente la codifica come valore numerico. Tuttavia, se il numero di valori univoci per la proprietà è inferiore al 10% del numero totale di valori e la cardinalità di tali valori univoci è inferiore a 100, Neptune ML utilizza una codifica categoriale.
- Se i valori delle proprietà possono essere convertiti in un tipo `datetime`, Neptune ML li codifica come funzionalità `datetime`.
- Se i valori delle proprietà possono essere impostati su valori booleani (1/0 o True/False), Neptune ML utilizza la codifica delle categorie.
- Se la proprietà è una stringa con valori univoci per più del 10% e numero medio di token per valore maggiore o uguale a 3, Neptune ML deduce che il tipo di proprietà è testo e rileva automaticamente la lingua utilizzata. Se la lingua rilevata è una di quelle supportate da fastText, ovvero inglese, cinese, hindi, spagnolo e francese, Neptune ML utilizza `text_fasttext` per codificare il testo. Altrimenti, Neptune ML utilizza `text_sbert`
- Se la proprietà è una stringa non classificata come funzionalità di testo, Neptune ML presume che sia una funzionalità categoriale e utilizza la codifica delle categorie.
- Se ogni nodo ha il proprio valore unico per una proprietà che viene dedotta come una funzionalità di categoria, Neptune ML elimina la proprietà dal grafo di addestramento perché probabilmente si tratta di un ID non informativo per l'apprendimento.
- Se è noto che la proprietà contiene separatori di Neptune validi come punto e virgola (";"), Neptune ML può trattare la proprietà solo come `MultiNumerical` o `MultiCategorical`.
  - Neptune ML tenta innanzitutto di codificare i valori come funzionalità numerica. Se ha esito positivo, Neptune ML utilizza la codifica numerica per creare funzionalità vettoriali numeriche.
  - In caso contrario, Neptune ML codifica i valori come multi-categoriali.
- Se Neptune ML non è in grado di dedurre il tipo di dati dei valori di una proprietà, Neptune ML elimina la proprietà dal grafo di addestramento.

Modifica di un file di configurazione dei dati di addestramento

Il processo di esportazione Neptune esporta i dati Neptune ML da un cluster Neptune DB in un bucket S3. Esporta nodi e archi separatamente in una cartella `nodes/` e in una cartella `edges/`. Crea inoltre un file di configurazione dei dati di addestramento JSON, denominato `training-data-configuration.json` per impostazione predefinita. Questo file contiene informazioni sullo schema del grafo, sui tipi delle relative funzionalità, sulle operazioni di trasformazione e normalizzazione delle funzionalità e sulla funzionalità di destinazione per un'attività di classificazione o regressione.

In alcuni casi, potrebbe essere necessario modificare direttamente il file di configurazione. Uno di questi casi si verifica quando si vuole cambiare il modo in cui le funzionalità vengono elaborate o il modo in cui viene costruito il grafo, senza dover eseguire nuovamente l'esportazione ogni volta che si desidera modificare le specifiche per l'attività di machine learning da risolvere.

Modifica di un file di configurazione dei dati di addestramento

1. Scaricare il file nel computer locale.

   A meno che non siano stati specificati uno o più processi denominati nel parametro `additionalParams/neptune_ml` passato al processo di esportazione, il file avrà il nome predefinito, ovvero `training-data-configuration.json`. Puoi usare un comando CLI AWS come questo per scaricare il file:

   ```
   aws s3 cp \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json \
     ./
   ```

2. Modifica il file tramite un editor di testo.

3. Carica il file modificato. Carica nuovamente il file modificato nella stessa posizione in Amazon S3 da cui lo hai scaricato, utilizzando un comando CLI AWS come questo:

   ```
   aws s3 cp \
     training-data-configuration.json \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json
   ```

Esempio di file di configurazione dei dati di addestramento JSON

Ecco un file di configurazione dei dati di addestramento di esempio che descrive un grafo per un'attività di classificazione dei nodi:

```
{
  "version" : "v2.0",
  "query_engine" : "gremlin",
  "graph" : [
    {
      "edges" : [
        {
          "file_name" : "edges/(movie)-included_in-(genre).csv",
          "separator" : ",",
          "source" : ["~from", "movie"],
          "relation" : ["", "included_in"],
          "dest" : [ "~to", "genre" ]
        },
        {
          "file_name" : "edges/(user)-rated-(movie).csv",
          "separator" : ",",
          "source" : ["~from", "movie"],
          "relation" : ["rating", "prefixname"], # [prefixname#value]
          "dest" : ["~to", "genre"],
          "features" : [
            {
              "feature" : ["rating", "rating", "numerical"],
              "norm" : "min-max"
            }
          ]
        }
      ],
      "nodes" : [
        {
          "file_name" : "nodes/genre.csv",
          "separator" : ",",
          "node" : ["~id", "genre"],
          "features" : [
            {
              "feature": ["name", "genre", "category"],
              "separator": ";"
            }
          ]
```

```
        },
        {
          "file_name" : "nodes/movie.csv",
          "separator" : ",",
          "node" : ["~id", "movie"],
          "features" : [
            {
              "feature": ["title", "title", "word2vec"],
              "language": ["en_core_web_lg"]
            }
          ]
        },
        {
          "file_name" : "nodes/user.csv",
          "separator" : ",",
          "node" : ["~id", "user"],
          "features" : [
            {
              "feature": ["age", "age", "numerical"],
              "norm" : "min-max",
              "imputation": "median",
            },
            {
              "feature": ["occupation", "occupation", "category"],
            }
          ],
          "labels" : [
            {
              "label": ["gender", "classification"],
              "split_rate" : [0.8, 0.2, 0.0]
            }
          ]
        }
      ]
    },
    "warnings" : [ ]
  ]
}
```

Struttura dei file di configurazione dei dati di addestramento JSON

Il file di configurazione di addestramento fa riferimento ai file CSV salvati dal processo di esportazione nelle cartelle `nodes/` e `edges/`.

In ogni file in `nodes/` sono archiviate informazioni sui nodi che hanno la stessa etichetta del nodo del grafo delle proprietà. Ogni colonna in un file dei nodi archivia l'ID del nodo o la proprietà del nodo. La prima riga del file contiene un'intestazione che specifica l'`~id` o il nome della proprietà per ogni colonna.

In ogni file in `edges/` sono archiviate informazioni sui nodi che hanno la stessa etichetta dell'arco del grafo delle proprietà. Ogni colonna in un file dei nodi archivia l'ID del nodo di origine, l'ID del nodo di destinazione o la proprietà dell'arco. La prima riga del file contiene un'intestazione che specifica `~from`, `~to` o il nome della proprietà per ogni colonna.

Il file di configurazione dei dati di addestramento contiene tre elementi di primo livello:

```
{
  "version" : "v2.0",
  "query_engine" : "gremlin",
  "graph" : [ ... ]
}
```

- `version` (stringa): versione del file di configurazione utilizzata.
- `query_engine` (stringa): il linguaggio di query utilizzato per esportare i dati del grafo. Attualmente è valido solo il linguaggio "gremlin".
- `graph` (array JSON): elenca uno o più oggetti di configurazione che contengono i parametri del modello per ciascuno dei nodi e degli archi che verranno utilizzati.

  Gli oggetti di configurazione nell'array del grafo hanno la struttura descritta nella sezione successiva.

Contenuto di un oggetto di configurazione elencato nell'array **`graph`**

Un oggetto di configurazione nell'array `graph` può contenere tre nodi di primo livello:

```
  {
    "edges"    : [ ... ],
    "nodes"    : [ ... ],
    "warnings" : [ ... ],
  }
```

- `edges` (array di oggetti JSON): ogni oggetto JSON specifica un set di parametri per definire come verrà trattato un arco del grafo durante l'elaborazione e l'addestramento del modello. Viene utilizzato solo con il motore Gremlin.
- `nodes` (array di oggetti JSON): ogni oggetto JSON specifica un set di parametri per definire come verrà trattato un nodo del grafo durante l'elaborazione e l'addestramento del modello. Viene utilizzato solo con il motore Gremlin.
- `warnings` (array di oggetti JSON): ogni oggetto contiene un avviso generato durante il processo di esportazione dei dati.

Contenuto di un oggetto di configurazione arco elencato nell'array **`edges`**

Un oggetto di configurazione arco elencato in un array `edges` può contenere i seguenti campi di primo livello:

```
    {
      "file_name" : "(path to a CSV file)",
      "separator" : "(separator character)",
      "source"    : ["(column label for starting node ID)", "(starting node type)"],
      "relation"  : ["(column label for the relationship name)", "(the prefix name
for the relationship name)"],
      "dest"      : ["(column label for ending node ID)", "(ending node type)"],
      "features"  : [(array of feature objects)],
      "labels"    : [(array of label objects)]
    }
```

- **`file_name`**: stringa che specifica il percorso di un file CSV in cui sono archiviate le informazioni sugli archi con la stessa etichetta del grafo delle proprietà.

  La prima riga del file contiene una riga di intestazione con le etichette di colonna.

  Le prime due etichette di colonna sono `~from` e `~to`. Nella prima colonna (colonna `~from`) è archiviato l'ID del nodo iniziale dell'arco e nella seconda (colonna `~to`) è archiviato l'ID del nodo finale dell'arco.

  Le etichette delle colonne rimanenti nella riga di intestazione specificano, per ogni colonna rimanente, il nome della proprietà dell'arco i cui valori sono stati esportati in quella colonna.
- **`separator`**: stringa contenente il delimitatore che separa le colonne nel file CSV.

- **`source`**: array JSON contenente due stringhe che specificano il nodo iniziale dell'arco. La prima stringa contiene il nome dell'intestazione della colonna in cui è archiviato l'ID del nodo iniziale. La seconda stringa specifica il tipo di nodo.
- **`relation`**: array JSON contenente due stringhe che specificano il tipo di relazione dell'arco. La prima stringa contiene il nome dell'intestazione della colonna in cui è archiviato il nome della relazione (`relname`). La seconda stringa contiene il prefisso per il nome della relazione (`prefixname`).

  Il tipo di relazione completo è costituito dalle due stringhe combinate con un trattino, in questo modo: *`prefixname`*`-`*`relname`*.

  Se la prima stringa è vuota, tutti gli archi hanno lo stesso tipo di relazione, ovvero la stringa `prefixname`.
- **`dest`**: array JSON contenente due stringhe che specificano il nodo finale dell'arco. La prima stringa contiene il nome dell'intestazione della colonna in cui è archiviato l'ID del nodo finale. La seconda stringa specifica il tipo di nodo.
- **`features`**: array JSON di oggetti funzionalità del valore della proprietà. Ogni oggetto funzionalità del valore della proprietà include i seguenti campi:
  - feature: array JSON di tre stringhe. La prima stringa contiene il nome dell'intestazione della colonna contenente il valore della proprietà. La seconda stringa contiene il nome della funzionalità. La terza stringa contiene il tipo di funzionalità.
  - norm (facoltativo): specifica un metodo di normalizzazione da applicare ai valori delle proprietà.

- **`labels`**: array di oggetti JSON. Ogni oggetto definisce una funzionalità di destinazione degli archi e specifica le proporzioni degli archi che le fasi di addestramento e convalida devono accettare. Ogni oggetto include i seguenti campi:
  - label: array JSON di due stringhe. La prima stringa contiene il nome dell'intestazione della colonna contenente il valore della proprietà della funzionalità di destinazione. La seconda stringa specifica uno dei seguenti tipi di attività di destinazione:
    - `"classification"`: attività di classificazione degli archi. I valori delle proprietà forniti nella colonna identificata dalla prima stringa nell'array `label` vengono trattati come valori categoriali. Per un'attività di classificazione degli archi, la prima stringa nell'array `label` non può essere vuota.
    - `"regression"`: attività di regressione degli archi. I valori delle proprietà forniti nella colonna identificata dalla prima stringa nell'array `label` vengono trattati come valori numerici. Per un'attività di regressione degli archi, la prima stringa nell'array `label` non può essere vuota.

    - `"link_prediction"`: attività di previsione dei collegamenti. Non sono richiesti valori delle proprietà. Per un'attività di previsione dei collegamenti, la prima stringa nell'array `label` viene ignorata.
- **`split_rate`**: array JSON contenente tre numeri compresi tra zero e uno che si sommano a uno e che rappresentano una stima delle proporzioni dei nodi che verranno utilizzati rispettivamente nelle fasi di addestramento, convalida e test. È possibile definire questo campo o `custom_split_filenames`, ma non entrambi. Vedi split_rate.
- **`custom_split_filenames`**: oggetto JSON che specifica i nomi dei file che definiscono le popolazioni di addestramento, convalida e test. È possibile definire questo campo o `split_rate`, ma non entrambi. Per ulteriori informazioni, consulta Proporzioni personalizzate per addestramento, convalida e test.

Contenuto di un oggetto di configurazione nodo elencato in un array **`nodes`**

Un oggetto di configurazione nodo elencato in un array `nodes` può contenere i seguenti campi:

```
    {
      "file_name" : "(path to a CSV file)",
      "separator" : "(separator character)",
      "node"      : ["(column label for the node ID)", "(node type)"],
      "features"  : [(feature array)],
      "labels"    : [(label array)],
    }
```

- **`file_name`**: stringa che specifica il percorso di un file CSV in cui sono archiviate le informazioni sui nodi con la stessa etichetta del grafo delle proprietà.

  La prima riga del file contiene una riga di intestazione con le etichette di colonna.

  L'etichetta della prima colonna è `~id` e nella prima colonna (colonna `~id`) è archiviato l'ID del nodo.

  Le etichette delle colonne rimanenti nella riga di intestazione specificano, per ogni colonna rimanente, il nome della proprietà del nodo i cui valori sono stati esportati in quella colonna.
- **`separator`**: stringa contenente il delimitatore che separa le colonne nel file CSV.
- **`node`**: array JSON contenente due stringhe. La prima stringa contiene il nome dell'intestazione della colonna in cui sono archiviati gli ID dei nodi. La seconda stringa specifica il tipo di nodo nel grafo, che corrisponde a un'etichetta del grafo delle proprietà del nodo.

- **`features`**: array JSON di oggetti funzionalità del nodo. Per informazioni, consultare Contenuto di un oggetto feature elencato in un array `features` per un nodo o un arco.
- **`labels`**: array JSON di oggetti etichetta del nodo. Per informazioni, consultare Contenuto di un oggetto label elencato in un array `labels` del nodo.

Contenuto di un oggetto feature elencato in un array **`features`** per un nodo o un arco

Un oggetto feature del nodo elencato in un array `features` può contenere i seguenti campi di primo livello:

- **`feature`**: array JSON di tre stringhe. La prima stringa contiene il nome dell'intestazione della colonna contenente il valore della proprietà della funzionalità. La seconda stringa contiene il nome della funzionalità.

  La terza stringa contiene il tipo di funzionalità. I tipi di funzionalità validi sono elencati in Valori possibili del campo `type` per le funzionalità.
- **`norm`**: campo obbligatorio per le funzioni numeriche. Specifica un metodo di normalizzazione da utilizzare sui valori numerici. I valori validi sono `"none"`, `"min-max"` e "standard". Per informazioni dettagliate, consulta Campo `norm`.
- **`language`**: il campo della lingua specifica la lingua utilizzata nei valori delle proprietà del testo. Il suo utilizzo dipende dal metodo di codifica del testo:
  - Per la codifica `text_fasttext` questo campo è obbligatorio e deve specificare una delle seguenti lingue:
    - `en` (inglese)
    - `zh` (cinese)
    - `hi` (hindi)
    - `es` (spagnolo)
    - `fr` (francese)

    Tuttavia, `text_fasttext` non può gestire più di una lingua alla volta.
  - Per la codifica `text_sbert` questo campo non viene utilizzato, poiché la codifica SBERT è multilingue.
  - Per la codifica `text_word2vec` questo campo è facoltativo, poiché `text_word2vec` supporta solo la lingua inglese. Se presente, deve specificare il nome del modello linguistico inglese:

```
"language" : "en_core_web_lg"
```

  - Per la codifica `tfidf` questo campo non viene utilizzato.

- **`max_length`**: questo campo è facoltativo per le funzionalità `text_fasttext`, in quanto specifica il numero massimo di token in una funzionalità di testo di input che verranno codificati. Il testo immesso una volta raggiunto il valore `max_length` viene ignorato. Ad esempio, l'impostazione di max_length su 128 indica che tutti i token successivi al 128° in una sequenza di testo verranno ignorati.
- **`separator`**: questo campo viene utilizzato facoltativamente con le funzionalità `category`, `numerical` e `auto`. Specifica un carattere che può essere utilizzato per suddividere il valore di una proprietà in più valori categoriali o valori numerici.

  Per informazioni, consultare Campo `separator`.
- **`range`**: questo campo è obbligatorio per le funzionalità `bucket_numerical`. Specifica l'intervallo di valori numerici che devono essere suddivisi in bucket.

  Per informazioni, consultare Campo `range`.
- **`bucket_cnt`**: questo campo è obbligatorio per le funzionalità `bucket_numerical`. Specifica il numero di bucket in cui deve essere suddiviso l'intervallo numerico definito dal parametro `range`.

  Per informazioni, consultare Funzionalità numeriche per bucket in Neptune ML.
- **`slide_window_size`**: questo campo viene usato facoltativamente con le funzionalità `bucket_numerical` per assegnare valori a più di un bucket:

  Per informazioni, consultare Campo `slide_window_size`.
- **`imputer`**: questo campo viene utilizzato facoltativamente con le funzionalità `numerical`, `bucket_numerical` e `datetime` per fornire una tecnica di imputazione per inserire i valori mancanti. Le tecniche di imputazione supportate sono `"mean"`, `"median"` e `"most_frequent"`.

  Per informazioni, consultare Campo `imputer`.
- **`max_features`**: questo campo viene utilizzato facoltativamente dalle funzionalità `text_tfidf` per specificare il numero massimo di termini da codificare.

  Per informazioni, consultare Campo `max_features`.
- **`min_df`**: questo campo viene utilizzato facoltativamente dalle funzionalità `text_tfidf` per specificare la frequenza minima nel documento dei termini da codificare.

Per informazioni, consultare Campo `min_df`.

- **`ngram_range`**: questo campo viene utilizzato facoltativamente dalle funzionalità `text_tfidf` per specificare le dimensioni delle sequenze di parole o token da considerare come potenziali termini individuali da codificare.

  Per informazioni, consultare Campo `ngram_range`.

- **`datetime_parts`**: questo campo viene utilizzato facoltativamente dalle funzionalità `datetime` per specificare quali parti del valore datetime codificare in modo categoriale.

  Per informazioni, consultare Campo `datetime_parts`.

Contenuto di un oggetto label elencato in un array **`labels`** del nodo

Un oggetto label elencato in un array `labels` del nodo definisce una funzionalità di destinazione del nodo e specifica le proporzioni dei nodi che verranno utilizzate nelle fasi di addestramento, convalida e test. Ogni oggetto può contenere i seguenti campi:

```
    {
      "label"      : ["(column label for the target feature property value)", "(task
type)"],
      "split_rate" : [(training proportion), (validation proportion), (test
proportion)],
      "custom_split_filenames" : {"train": "(training file name)", "valid":
"(validation file name)", "test": "(test file name)"},
      "separator"  : "(separator character for node-classification category values)",
    }
```

- **`label`**: array JSON contenente due stringhe. La prima stringa contiene il nome dell'intestazione della colonna in cui sono archiviati i valori delle proprietà per la funzionalità. La seconda stringa specifica il tipo di attività di destinazione, che può essere:
  - `"classification"`: attività di classificazione dei nodi. I valori delle proprietà nella colonna specificata vengono utilizzati per creare una funzionalità categoriale.
  - `"regression"`: attività di regressione dei nodi. I valori delle proprietà nella colonna specificata vengono utilizzati per creare una funzionalità numerica.
- **`split_rate`**: array JSON contenente tre numeri compresi tra zero e uno che si sommano a uno e che rappresentano una stima delle proporzioni dei nodi che verranno utilizzati rispettivamente nelle fasi di addestramento, convalida e test. Per informazioni, consultare split_rate.

- **`custom_split_filenames`**: oggetto JSON che specifica i nomi dei file che definiscono le popolazioni di addestramento, convalida e test. È possibile definire questo campo o `split_rate`, ma non entrambi. Per ulteriori informazioni, consulta Proporzioni personalizzate per addestramento, convalida e test.
- **`separator`**: stringa contenente il delimitatore che separa i valori delle funzionalità categoriali per un'attività di classificazione.

> **Note**
>
> Se non viene specificato alcun oggetto etichetta sia per gli archi che per i nodi, si presume automaticamente che si tratti di un'attività di previsione dei collegamenti e gli archi vengono divisi casualmente in 90% per l'addestramento e 10% per la convalida.

Proporzioni personalizzate per addestramento, convalida e test

Per impostazione predefinita, Neptune ML utilizza il parametro `split_rate` per dividere il grafo in modo casuale in popolazioni di addestramento, convalida e test utilizzando le proporzioni definite in questo parametro. Per avere un controllo più preciso sulle entità utilizzate in queste diverse popolazioni, è possibile creare file che le definiscono in modo esplicito, quindi modificare il file di configurazione dei dati di addestramento per mappare questi file di indicizzazione alle popolazioni. Questa mappatura è specificata da un oggetto JSON per la chiave `custom_split_filesnames` nel file di configurazione dell'addestramento. Se viene utilizzata questa opzione, è obbligatorio specificare il nome del file per le chiavi `train` e `validation`, mentre è facoltativo per la chiave `test`.

La formattazione di questi file deve corrispondere al formato dei dati Gremlin. In particolare, per le attività a livello di nodo, ogni file deve contenere una colonna con l'intestazione `~id` che elenca gli ID dei nodi, mentre per le attività a livello di arco, i file devono specificare `~from` e `~to` per indicare rispettivamente i nodi di origine e di destinazione degli archi. Questi file devono essere collocati nella stessa posizione Amazon S3 dei dati esportati utilizzati per l'elaborazione dei dati (consulta: `outputS3Path`).

Per le attività di classificazione o regressione delle proprietà, questi file possono facoltativamente definire le etichette per l'attività di machine learning. In tal caso, i file devono avere una colonna delle proprietà con lo stesso nome di intestazione definito nel file di configurazione dei dati di

addestramento. Se le etichette delle proprietà sono definite sia nei file dei nodi e degli archi esportati che nei file di divisione personalizzata, i file di divisione personalizzata avranno la priorità.

Addestramento di un modello con Neptune ML.

Dopo aver esportato da Neptune i dati da utilizzare per l'addestramento dei modelli, puoi avviare un processo di elaborazione dei dati utilizzando un comando `curl` (o `awscurl`) come il seguente:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
      }'
```

I dettagli su come utilizzare questo comando sono illustrati in Comando modeltraining, insieme a informazioni su come recuperare lo stato di un processo in esecuzione, come arrestare un processo in esecuzione e come elencare tutti i processi in esecuzione.

È inoltre possibile fornire un `previousModelTrainingJobId` per usare le informazioni provenienti dal processo di addestramento di un modello Neptune ML completato per accelerare la ricerca degli iperparametri in un nuovo processo di addestramento. Questo è utile durante il riaddestramento del modello su nuovi dati del grafo, nonché per l'addestramento incrementale sugli stessi dati del grafo. Usa un comando come questo:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "previousModelTrainingJobId" : "(the model-training job-id of a completed job)"
      }'
```

Puoi addestrare l'implementazione di un modello personalizzato sull'infrastruttura di addestramento di Neptune ML specificando un oggetto `customModelTrainingParameters`, in questo modo:

```
curl \
```

```
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Per ulteriori informazioni, ad esempio su come recuperare lo stato di un processo in esecuzione, su come arrestare un processo in esecuzione e su come elencare tutti i lavori in esecuzione, consulta Comando modeltraining. Per ulteriori informazioni su come implementare e usare un modello personalizzato, consulta Modelli personalizzati in Neptune ML..

Argomenti

- Modelli e addestramento dei modelli in Amazon Neptune ML
- Personalizzazione delle configurazioni degli iperparametri dei modelli in Neptune ML
- Best practice per l'addestramento dei modelli

Modelli e addestramento dei modelli in Amazon Neptune ML

Neptune ML utilizza reti neurali a grafo (GNN) per creare modelli per le varie attività di machine learning. È stato dimostrato che le reti neurali a grafo ottengono risultati all'avanguardia per le attività di machine learning a grafo e sono eccellenti nell'estrazione di modelli informativi da dati strutturati a grafo.

Reti neurali a grafo (GNN) in Neptune ML

Le reti neurali a grafo (GNN) appartengono a una famiglia di reti neurali che calcolano le rappresentazioni dei nodi tenendo conto della struttura e delle funzionalità dei nodi vicini. Le GNN si aggiungono ad altri metodi tradizionali di machine learning e delle rete neurali che non sono adatti per i dati a grafo.

Le GNN consentono di risolvere attività di machine learning come la classificazione e la regressione dei nodi (per prevedere le proprietà dei nodi), la classificazione e la regressione degli archi (per prevedere le proprietà degli archi) o la previsione dei collegamenti (per prevedere se due nodi nel grafo devono essere connessi o meno).

In generale, l'utilizzo di una GNN per un'attività di machine learning prevede due fasi:

- Una fase di codifica, in cui la GNN calcola un vettore d-dimensionale per ogni nodo del grafo. Questi vettori sono anche chiamati rappresentazioni o incorporamenti.
- Una fase di decodifica, che effettua previsioni basate sulle rappresentazioni codificate.

Per la classificazione e la regressione dei nodi, le rappresentazioni dei nodi vengono utilizzate direttamente per le attività di classificazione e regressione. Per la classificazione e la regressione degli archi, le rappresentazioni dei nodi incidenti su un arco vengono utilizzate come input per la classificazione o la regressione. Per la previsione dei collegamenti, viene calcolato un punteggio di probabilità degli archi utilizzando una coppia di rappresentazioni di nodi e una rappresentazione del tipo di arco.

La Deep Graph Library (DGL) facilita la definizione e l'addestramento efficienti delle GNN per queste attività.

Diversi modelli di GNN sono unificati sotto la formulazione del passaggio dei messaggi. Da questo punto di vista, la rappresentazione di un nodo in un grafo viene calcolata utilizzando le rappresentazioni dei vicini del nodo (i messaggi), insieme alla rappresentazione iniziale del nodo. In

Neptune ML la rappresentazione iniziale di un nodo deriva dalle funzionalità estratte dalle proprietà del nodo oppure è apprendibile e dipende dall'identità del nodo.

Neptune ML consente inoltre di concatenare le funzionalità dei nodi e le rappresentazioni dei nodi apprendibili per fungere da rappresentazione del nodo originale.

Per le varie attività di Neptune ML che coinvolgono grafi con proprietà dei nodi, viene usata la rete Relational Graph Convolutional Network (R-GCN) per eseguire la fase di codifica. La R-GCN è un'architettura GNN adatta per i grafi con più tipi di nodi e archi (noti come grafi eterogenei).

La rete R-GCN è costituita da un numero fisso di livelli, impilati uno dopo l'altro. Ogni livello della rete R-GCN utilizza i relativi parametri del modello apprendibili per aggregare informazioni dal neighborhood a 1 hop di un nodo. Poiché i livelli successivi utilizzano le rappresentazioni di output del livello precedente come input, il raggio del neighborhood del grafo che influenza l'incorporamento finale di un nodo dipende dal numero di livelli (`num-layer`) della rete R-GCN.

Questo significa ad esempio che una rete a 2 livelli utilizza le informazioni provenienti da nodi che si trovano a 2 hop di distanza.

Per ulteriori informazioni sulle reti GNN, consulta A Comprehensive Survey on Graph Neural Networks. Per ulteriori informazioni sulla Deep Graph Library (DGL), visita la pagina web di DGL. Per un tutorial pratico sull'uso della DGL con le reti GNN, consulta Learning graph neural networks with Deep Graph Library.

Addestramento delle reti neurali a grafo

Nel machine learning il processo che consente a un modello di apprendere come fare previsioni valide per un'attività è denominato addestramento del modello. Questa operazione viene in genere eseguita specificando un obiettivo particolare da ottimizzare e un algoritmo da utilizzare per eseguire tale ottimizzazione.

Questo processo viene impiegato anche per addestrare una rete GNN ad apprendere rappresentazioni valide anche per l'attività a valle. Per tale attività viene creata una funzione obiettivo che viene ridotta al minimo durante l'addestramento del modello. Ad esempio, per la classificazione dei nodi si usa CrossEntropyLoss come obiettivo per penalizzare le classificazioni errate, mentre per la regressione dei nodi si riduce al minimo MeanSquareError.

L'obiettivo è in genere una funzione di perdita che accetta le previsioni del modello per un punto dati specifico e le confronta con il valore ground-truth per tale punto dati. Restituisce il valore di perdita,

che mostra quanto siano lontane le previsioni del modello. L'obiettivo del processo di addestramento è ridurre al minimo le perdite e garantire che le previsioni dei modelli siano vicine al valore ground-truth.

L'algoritmo di ottimizzazione utilizzato nel deep learning per il processo di addestramento è in genere una variante della discesa del gradiente. In Neptune ML, utilizziamo Adam, un algoritmo per l'ottimizzazione basata su gradiente del primo ordine di funzioni obiettivo stocastiche basate su stime adattive di momenti di ordine inferiore.

Sebbene il processo di addestramento del modello tenti di garantire che i parametri del modello appresi siano vicini ai minimi della funzione obiettivo, le prestazioni complessive di un modello dipendono anche dagli iperparametri del modello, che sono impostazioni del modello che non vengono apprese dall'algoritmo di addestramento. Ad esempio, la dimensionalità della rappresentazione del nodo appresa, `num-hidden`, è un iperparametro che influisce sulle prestazioni del modello. Pertanto, nel machine learning è comune eseguire l'ottimizzazione degli iperparametri per scegliere gli iperparametri adatti.

Neptune ML utilizza un processo di ottimizzazione degli iperparametri di SageMaker per avviare più istanze di addestramento del modello con diverse configurazioni di iperparametri per tentare di trovare il modello migliore per una serie di impostazioni di iperparametri. Per informazioni, consultare Personalizzazione delle configurazioni degli iperparametri dei modelli in Neptune ML.

Modelli di incorporamento del grafo della conoscenza in Neptune ML

I grafi della conoscenza sono grafi che codificano informazioni su diverse entità (nodi) e le relative relazioni (archi). In Neptune ML i modelli di incorporamento del grafo della conoscenza vengono applicati per impostazione predefinita per eseguire la previsione dei collegamenti quando il grafo non contiene proprietà dei nodi, ma solo relazioni con altri nodi. Tuttavia, anche i modelli R-GCN con incorporamenti apprendibili possono essere utilizzati per questi grafi specificando il tipo di modello `"rgcn"`, poiché i modelli di incorporamento dei grafi della conoscenza sono più semplici e progettati per essere efficaci per l'apprendimento di rappresentazioni per grafi della conoscenza su larga scala.

I modelli di incorporamento dei grafi della conoscenza vengono usati in un'attività di previsione dei collegamenti per prevedere i nodi o le relazioni che completano una tripla `(h, r, t)`, dove **`h`** è il nodo di origine, **`r`** è il tipo di relazione e **`t`** è il nodo di destinazione.

I modelli di incorporamento del grafo della conoscenza implementati in Neptune ML sono `distmult`, `transE` e `rotatE`. Per ulteriori informazioni sui modelli di incorporamento del grafo della conoscenza, consulta DGL-KE.

Addestramento di modelli personalizzati in Neptune ML.

Neptune ML consente di definire e implementare modelli personalizzati, per scenari specifici. Per informazioni su come implementare un modello personalizzato e su come utilizzare l'infrastruttura di Neptune ML per addestrarlo, consulta Modelli personalizzati in Neptune ML..

Personalizzazione delle configurazioni degli iperparametri dei modelli in Neptune ML

Quando si avvia un processo di addestramento dei modelli Neptune ML, Neptune ML utilizza automaticamente le informazioni dedotte dal precedente processo di elaborazione dei dati. Usa le informazioni per generare intervalli di configurazione degli iperparametri che vengono utilizzati per creare un processo di ottimizzazione degli iperparametri di SageMaker per addestrare più modelli per l'attività. In questo modo, non è necessario specificare un lungo elenco di valori di iperparametri da usare per l'addestramento dei modelli. Gli intervalli e i valori predefiniti degli iperparametri del modello vengono invece selezionati in base al tipo di attività, al tipo di grafo e alle impostazioni del processo di ottimizzazione.

Tuttavia, è anche possibile ignorare la configurazione degli iperparametri predefinita e specificare iperparametri personalizzati modificando un file di configurazione JSON generato dal processo di elaborazione dei dati.

Utilizzando l'API modelTraining di Neptune ML, puoi controllare diverse impostazioni generali degli iperparametri del processo di ottimizzazione, come `maxHPONumberOfTrainingJobs`, `maxHPOParallelTrainingJobs` e `trainingInstanceType`. Per un controllo più granulare sugli iperparametri del modello, è possibile personalizzare il file `model-HPO-configuration.json` generato dal processo di elaborazione dei dati. Il file viene salvato nella posizione Amazon S3 specificata per l'output del processo di elaborazione.

Puoi scaricare il file, modificarlo per sovrascrivere le configurazioni degli iperparametri predefinite e caricarlo nuovamente nella stessa posizione Amazon S3. Non modificare il nome del file e presta attenzione a seguire queste istruzioni durante la modifica.

Per scaricare il file da S3

```
aws s3 cp \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json \
  ./
```

Al termine della modifica, carica nuovamente il file nella posizione originale:

```
aws s3 cp \
  model-HPO-configuration.json \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json
```

Struttura del file **model-HPO-configuration.json**

Il file `model-HPO-configuration.json` specifica il modello da addestrare, il tipo di attività `task_type` di machine learning e gli iperparametri che devono essere variati o fissi per le varie esecuzioni di addestramento del modello.

Gli iperparametri sono classificati come appartenenti a vari livelli che indicano la precedenza assegnata agli iperparametri quando viene richiamato il processo di ottimizzazione degli iperparametri:

- Gli iperparametri di livello 1 hanno la precedenza più alta. Se imposti `maxHPONumberOfTrainingJobs` su un valore minore di 10, vengono ottimizzati solo gli iperparametri di livello 1 e gli altri assumono i valori predefiniti.
- Gli iperparametri di livello 2 hanno una precedenza inferiore, quindi se sono presenti più di 10 ma meno di 50 processi di addestramento totali per un processo di ottimizzazione, vengono ottimizzati sia gli iperparametri di livello 1 che di livello 2.
- Gli iperparametri di livello 3 vengono ottimizzati insieme a quelli di livello 1 e 2 solo se sono presenti più di 50 processi di addestramento totali.
- Infine, gli iperparametri fissi non vengono mai ottimizzati e assumono sempre i valori predefiniti.

Esempio di file **model-HPO-configuration.json**

Di seguito è riportato un file `model-HPO-configuration.json` di esempio.

```
{
  "models": [
    {
      "model": "rgcn",
      "task_type": "node_class",
      "eval_metric": {
        "metric": "acc"
      },
      "eval_frequency": {
          "type":  "evaluate_every_epoch",
          "value":  1
      },
      "1-tier-param": [
        {
            "param": "num-hidden",
            "range": [16, 128],
```

```
            "type": "int",
            "inc_strategy": "power2"
        },
        {
          "param": "num-epochs",
          "range": [3,30],
          "inc_strategy": "linear",
          "inc_val": 1,
          "type": "int",
          "node_strategy": "perM"
        },
        {
          "param": "lr",
          "range": [0.001,0.01],
          "type": "float",
          "inc_strategy": "log"
        }
      ],
      "2-tier-param": [
        {
          "param": "dropout",
          "range": [0.0,0.5],
          "inc_strategy": "linear",
          "type": "float",
          "default": 0.3
        },
        {
          "param": "layer-norm",
          "type": "bool",
          "default": true
        }
      ],
      "3-tier-param": [
        {
          "param": "batch-size",
          "range": [128, 4096],
          "inc_strategy": "power2",
          "type": "int",
          "default": 1024
        },
        {
          "param": "fanout",
          "type": "int",
          "options": [[10, 30],[15, 30], [15, 30]],
```

```
          "default": [10, 15, 15]
        },
        {
          "param": "num-layer",
          "range": [1, 3],
          "inc_strategy": "linear",
          "inc_val": 1,
          "type": "int",
          "default": 2
        },
        {
          "param": "num-bases",
          "range": [0, 8],
          "inc_strategy": "linear",
          "inc_val": 2,
          "type": "int",
          "default": 0
        }
      ],
      "fixed-param": [
        {
          "param": "concat-node-embed",
          "type": "bool",
          "default": true
        },
        {
          "param": "use-self-loop",
          "type": "bool",
          "default": true
        },
        {
          "param": "low-mem",
          "type": "bool",
          "default": true
        },
        {
          "param": "l2norm",
          "type": "float",
          "default": 0
        }
      ]
    }
  ]
```

```
}
```

Elementi di un file **`model-HPO-configuration.json`**

Il file contiene un oggetto JSON con un singolo array di primo livello denominato `models` che contiene un singolo oggetto configurazione del modello. Quando personalizzi il file, assicurati che l'array `models` contenga solo un oggetto configurazione del modello. Se il file contiene più di un oggetto configurazione del modello, il processo di ottimizzazione avrà esito negativo e verrà visualizzato un avviso.

L'oggetto configurazione del modello contiene i seguenti elementi di primo livello:

- **`model`** (stringa): tipo di modello da addestrare (da non modificare). I valori validi sono:
  - `"rgcn"`: impostazione predefinita per le attività di classificazione e regressione dei nodi e per le attività di previsione dei collegamenti eterogenee.
  - `"transe"`: impostazione predefinita per le attività di previsione dei collegamenti di incorporamento del grafo della conoscenza.
  - `"distmult"`: tipo di modello alternativo per le attività di previsione dei collegamenti di incorporamento del grafo della conoscenza.
  - `"rotate"`: tipo di modello alternativo per le attività di previsione dei collegamenti di incorporamento del grafo della conoscenza.

  In generale, è consigliabile non modificare direttamente il valore `model`, in quanto diversi tipi di modello hanno spesso iperparametri applicabili sostanzialmente diversi e ciò può causare un errore di analisi dopo l'avvio del processo di addestramento.

  Per modificare il tipo di modello, usa il parametro `modelName` nell'API modelTraining anziché modificarlo nel file `model-HPO-configuration.json`.

  Un modo per modificare il tipo di modello e apportare modifiche granulari agli iperparametri consiste nel copiare il modello di configurazione del modello predefinito per il modello da usare e incollarlo nel file `model-HPO-configuration.json`. Esiste una cartella denominata `hpo-configuration-templates` nella stessa posizione Amazon S3 del file `model-HPO-configuration.json` se il tipo di attività dedotto supporta più modelli. Questa cartella contiene tutte le configurazioni di iperparametri predefinite per gli altri modelli applicabili all'attività.

  Ad esempio, per modificare le configurazioni del modello e degli iperparametri per un'attività di previsione dei collegamenti KGE dal modello predefinito `transe` a un modello `distmult`, è

sufficiente incollare il contenuto del file `hpo-configuration-templates/distmult.json` nel file `model-HPO-configuration.json` e quindi modificare gli iperparametri in base alle esigenze.

> **Note**
>
> Se imposti il parametro `modelName` nell'API `modelTraining` e modifichi il `model` e la specifica degli iperparametri nel file `model-HPO-configuration.json` e questi sono diversi, il valore `model` nel file `model-HPO-configuration.json` ha la precedenza e il valore `modelName` viene ignorato.

- **`task_type`** (stringa): tipo di attività di machine learning dedotto o passato direttamente al processo di elaborazione dei dati (da non modificare). I valori validi sono:
  - `"node_class"`
  - `"node_regression"`
  - `"link_prediction"`

  Il processo di elaborazione dei dati deduce il tipo di attività esaminando il set di dati esportato e il file di configurazione del processo di apprendimento generato per verificare le proprietà del set di dati.

  Questo valore non deve essere modificato. Se si desidera addestrare un'attività diversa, è necessario eseguire un nuovo processo di elaborazione dei dati. Se il valore `task_type` non è quello previsto, è consigliabile verificare gli input del processo di elaborazione dati per assicurarsi che siano corretti. Sono inclusi i parametri nell'API `modelTraining` e il file di configurazione del processo di addestramento generato dal processo di esportazione dei dati.
- **`eval_metric`** (stringa): la metrica di valutazione deve essere utilizzata per valutare le prestazioni del modello e per selezionare il modello con le migliori prestazioni tra le esecuzioni di ottimizzazione degli iperparametri. I valori validi sono:
  - `"acc"`: accuratezza della classificazione standard. Si tratta dell'impostazione predefinita per le attività di classificazione a etichetta singola, a meno che non vengano rilevate etichette sbilanciate durante l'elaborazione dei dati, nel qual caso l'impostazione predefinita è `"F1"`.
  - `"acc_topk"`: numero di volte in cui l'etichetta corretta compare tra le prime **k** previsioni. Puoi anche impostare il valore **k** passando `topk` come chiave aggiuntiva.
  - `"F1"`: punteggio F1.
  - `"mse"`: metrica dell'errore quadratico medio, per le attività di regressione.

- `"mrr"`: metrica della classificazione reciproca media.
- `"precision"`: precisione del modello, calcolata come rapporto tra veri positivi e positivi previsti: `= true-positives / (true-positives + false-positives)`.
- `"recall"`: sensibilità del modello, calcolata come rapporto tra veri positivi e positivi effettivi: `= true-positives / (true-positives + false-negatives)`.
- `"roc_auc"`: area sotto la curva ROC. Questa è l'impostazione predefinita per la classificazione multi-etichetta.

Ad esempio, per modificare la metrica in `F1`, modifica il valore `eval_metric` come indicato di seguito:

```
"  eval_metric": {
    "metric": "F1",
  },
```

In alternativa, per modificare la metrica in un punteggio di accuratezza `topk`, dovrai modificare `eval_metric` come indicato di seguito:

```
  "eval_metric": {
    "metric": "acc_topk",
    "topk": 2
  },
```

- **`eval_frequency`** (oggetto): specifica con quale frequenza durante l'addestramento devono essere verificate le prestazioni del modello sul set di convalida. In base alle prestazioni di convalida, è quindi possibile avviare l'arresto anticipato e salvare il modello migliore.

  L'oggetto `eval_frequency` contiene due elementi, ovvero `"type"` e `"value"`. Ad esempio:

```
  "eval_frequency": {
    "type":  "evaluate_every_pct",
    "value":  0.1
  },
```

  I valori `type` validi sono:

  - **`evaluate_every_pct`**: specifica la percentuale di addestramento da completare per ogni valutazione.

Per `evaluate_every_pct`, il campo `"value"` contiene un numero a virgola mobile compreso tra zero e uno che esprime tale percentuale.

- **`evaluate_every_batch`**: specifica il numero di batch di addestramento da completare per ogni valutazione.

  Per `evaluate_every_batch`, il campo `"value"` contiene un numero intero che esprime il numero di batch.

- **`evaluate_every_epoch`**: specifica il numero di epoche per valutazione, dove una nuova epoca inizia a mezzanotte.

  Per `evaluate_every_epoch`, il campo `"value"` contiene un numero intero che esprime il numero di epoche.

L'impostazione predefinita per `eval_frequency` è:

```
"eval_frequency": {
  "type":  "evaluate_every_epoch",
  "value":  1
},
```

- **`1-tier-param`** (obbligatorio): array di iperparametri di livello 1.

  Se non si desidera ottimizzare alcun iperparametro, è possibile impostare questo parametro su un array vuoto. Questo non influisce sul numero totale di processi di addestramento avviati dal processo di ottimizzazione degli iperparametri di SageMaker. Significa solo che tutti i processi di addestramento, se ne esistono più di 1 ma meno di 10, verranno eseguiti con lo stesso set di iperparametri.

  D'altra parte, per trattare tutti gli iperparametri ottimizzabili con uguale importanza, puoi includere tutti gli iperparametri in questo array.

- **`2-tier-param`** (obbligatorio): array di iperparametri di livello 2.

  Questi parametri vengono ottimizzati solo se il valore di `maxHPONumberOfTrainingJobs` è maggiore di 10. In caso contrario, assumono i valori predefiniti.

  Se disponi di un budget di addestramento di almeno 10 processi di addestramento o se non desideri iperparametri di livello 2 per altri motivi, ma desideri ottimizzare tutti gli iperparametri ottimizzabili, puoi impostare questo parametro su un array vuoto.

- **`3-tier-param`** (obbligatorio): array di iperparametri di livello 3.

  Questi parametri vengono ottimizzati solo se il valore di `maxHPONumberOfTrainingJobs` è maggiore di 50. In caso contrario, assumono i valori predefiniti.

  Se non si desiderano iperparametri di livello 3, è possibile impostare questo parametro su un array vuoto.
- **`fixed-param`** (obbligatorio): array di iperparametri fissi che accettano solo i relativi valori predefiniti e non variano in processi di addestramento diversi.

  Per variare tutti gli iperparametri, puoi impostare questo parametro su un array vuoto e impostare il valore per `maxHPONumberOfTrainingJobs` su un numero sufficientemente elevato per variare tutti i livelli o impostare tutti gli iperparametri come di livello 1.

L'oggetto JSON che rappresenta ogni iperparametro in `1-tier-param`, `2-tier-param`, `3-tier-param` e `fixed-param` contiene i seguenti elementi:

- **`param`** (stringa): nome dell'iperparametro (da non modificare).

  Vedi l'elenco dei nomi di iperparametri validi in Neptune ML.
- **`type`** (stringa): tipo dell'iperparametro (da non modificare).

  I tipi validi sono `bool`, `int` e `float`.
- **`default`** (stringa): valore predefinito per l'iperparametro.

  È possibile impostare un nuovo valore predefinito.

Gli iperparametri ottimizzabili possono contenere anche i seguenti elementi:

- **`range`** (array): intervallo per un iperparametro ottimizzabile in modo continuo.

  Deve essere un array con due valori, ovvero il valore minimo e il valore massimo dell'intervallo (`[min, max]`).
- **`options`** (array): opzioni per un iperparametro categoriale ottimizzabile.

  Questo array deve contenere tutte le opzioni da considerare:

```
"options" : [value1, value2, ... valuen]
```

- **`inc_strategy`** (stringa): tipo di modifica incrementale per gli intervalli di iperparametri ottimizzabili in modo continuo (da non modificare).

  I valori validi sono `log`, `linear` e `power2`. Si applica solo quando è impostata la chiave di intervallo.

  La modifica di questo elemento può determinare il mancato utilizzo dell'intero intervallo dell'iperparametro per l'ottimizzazione.
- **`inc_val`** (float): quantità in base alla differiscono gli incrementi successivi per gli iperparametri ottimizzabili in modo continuo (da non modificare).

  Si applica solo quando è impostata la chiave di intervallo.

  La modifica di questo elemento può determinare il mancato utilizzo dell'intero intervallo dell'iperparametro per l'ottimizzazione.
- **`node_strategy`** (stringa): indica che l'intervallo effettivo per questo iperparametro deve cambiare in base al numero di nodi nel grafo (da non modificare).

  I valori validi sono `"perM"` (per milione), `"per10M"` (per 10 milioni) e `"per100M"` (per 100 milioni).

  Invece di modificare questo valore, prova a modificare `range`.
- **`edge_strategy`** (stringa): indica che l'intervallo effettivo per questo iperparametro deve cambiare in base al numero di archi nel grafo (da non modificare).

  I valori validi sono `"perM"` (per milione), `"per10M"` (per 10 milioni) e `"per100M"` (per 100 milioni).

  Invece di modificare questo valore, prova a modificare `range`.

Elenco di tutti gli iperparametri in Neptune ML

L'elenco seguente contiene tutti gli iperparametri che possono essere impostati ovunque in Neptune ML, per qualsiasi tipo di modello e attività. Poiché non sono tutti applicabili a ogni tipo di modello, è importante impostare nel file `model-HPO-configuration.json` solo gli iperparametri disponibili nel modello in uso.

- **`batch-size`**: dimensioni del batch dei nodi di destinazione utilizzati in un unico passaggio in avanti. Tipo: `int`.

L'impostazione di questo elemento su un valore molto elevato può causare problemi di memoria per l'addestramento sulle istanze GPU.

- **`concat-node-embed`**: indica se ottenere o meno la rappresentazione iniziale di un nodo concatenando le relative funzionalità elaborate con incorporamenti iniziali del nodo apprendibili per aumentare l'espressività del modello. Tipo: `bool`.
- **`dropout`**: probabilità di abbandono applicata ai livelli di abbandono. Tipo: `float`.
- **`edge-num-hidden`**: dimensioni del livello nascosto o il numero di unità per il modulo della funzionalità arco. Usato solo quando `use-edge-features` è impostato su `True`. Tipo: float.
- **`enable-early-stop`**: attiva/disattiva l'utilizzo della funzionalità di arresto anticipato. Tipo: `bool`. Default: `true`

  Usa questo parametro booleano per disattivare la funzionalità di arresto anticipato.
- **`fanout`**: numero di neighbor da campionare per un nodo di destinazione durante il campionamento dei neighbor. Tipo: `int`.

  Questo valore è strettamente associato a `num-layers` e deve trovarsi sempre nello stesso livello di iperparametri. Questo perché puoi specificare un fanout per ogni potenziale livello GNN.

  Poiché questo iperparametro può causare ampie variazioni nelle prestazioni del modello, deve essere fisso o impostato come iperparametro di livello 2 o di livello 3. L'impostazione di questo elemento su un valore elevato può causare problemi di memoria per l'addestramento sulle istanze GPU.
- **`gamma`**: valore del margine nella funzione di punteggio. Tipo: `float`.

  Si applica solo ai modelli di previsione dei collegamenti `KGE`.
- **`l2norm`**: valore di decadimento del peso utilizzato nell'ottimizzatore che impone una penalità di normalizzazione L2 ai pesi. Tipo: `bool`.
- **`layer-norm`**: indica se utilizzare la normalizzazione dei livelli per i modelli `rgcn`. Tipo: `bool`.
- **`low-mem`**: indica se utilizzare un'implementazione a bassa memoria della funzione di passaggio dei messaggi di relazione a scapito della velocità. Tipo: `bool`.
- **`lr`**: velocità di apprendimento. Tipo: `float`.

  Deve essere impostato come un iperparametro di primo livello.

- **`neg-share`**: nella previsione dei collegamenti, indica se gli archi campionati positivi possono condividere campioni di archi negativi. Tipo: `bool`.
- **`num-bases`**: numero di basi per la scomposizione delle basi in un modello `rgcn`. L'utilizzo di un valore `num-bases` inferiore al numero di tipi di archi nel grafo funge da regolarizzatore per il modello `rgcn`. Tipo: `int`.
- **`num-epochs`**: numero di epoche di addestramento da eseguire. Tipo: `int`.

  Un'epoca è un passaggio di addestramento completo attraverso il grafo.
- **`num-hidden`**: dimensioni del livello nascosto o numero di unità. Tipo: `int`.

  Questo elemento imposta anche le dimensioni dell'incorporamento iniziale per i nodi senza funzionalità.

  L'impostazione di questo elemento su un valore molto elevato senza la riduzione di `batch-size` può causare problemi di memoria per l'addestramento sulle istanze GPU.
- **`num-layer`**: numero di livelli GNN nel modello. Tipo: `int`.

  Questo valore è strettamente associato al parametro fanout e deve essere posizionato dopo aver impostato il fanout nello stesso livello di iperparametri.

  Poiché questo iperparametro può causare ampie variazioni nelle prestazioni del modello, deve essere fisso o impostato come iperparametro di livello 2 o di livello 3.
- **`num-negs`**: nella previsione dei collegamenti, il numero di campioni negativi per campione positivo. Tipo: `int`.
- **`per-feat-name-embed`**: indica se incorporare ogni funzionalità trasformandola in modo indipendente prima di combinare le funzionalità. Tipo: `bool`.

  Se questo elemento è impostato su `true`, ogni funzionalità per nodo viene trasformata in modo indipendente in in dimensioni di dimensione fissa prima che tutte le funzionalità trasformate per il nodo vengano concatenate e ulteriormente trasformate nella dimensione `num_hidden`.

  Se questo elemento è impostato su `false`, le funzionalità vengono concatenate senza trasformazioni specifiche delle funzionalità.
- **`regularization-coef`**: nella previsione dei collegamenti, il coefficiente di perdita di regolarizzazione. Tipo: `float`.
- **`rel-part`**: indica se utilizzare la partizione di relazione per la previsione dei collegamenti `KGE`. Tipo: `bool`.

- **`sparse-lr`**: velocità di apprendimento per gli incorporamenti di nodi apprendibili. Tipo: `float`.

  Gli incorporamenti iniziali dei nodi apprendibili vengono utilizzati per i nodi senza funzionalità o quando è impostato il parametro `concat-node-embed`. I parametri del livello di incorporamento dei nodi apprendibili sparse vengono addestrati utilizzando un ottimizzatore separato che può avere una velocità di apprendimento distinta.
- **`use-class-weight`**: indica se applicare i pesi di classe per attività di classificazione non bilanciate. Se impostato su `true`, viene usato il numero di etichette per impostare un peso per ogni etichetta di classe. Tipo: `bool`.
- **`use-edge-features`**: indica se utilizzare le funzionalità degli archi durante il passaggio dei messaggi. Se impostato su `true`, viene aggiunto un modulo di funzionalità degli archi personalizzato al livello RGCN per i tipi di archi che dispongono di funzionalità. Tipo: `bool`.
- **`use-self-loop`**: indica se includere i cappi nell'addestramento di un modello `rgcn`. Tipo: `bool`.
- **`window-for-early-stop`**: controlla il numero degli ultimi punteggi di convalida in base a cui calcolare la media per decidere se effettuare un arresto anticipato. Il valore predefinito è 3. type=int. Consulta anche Arresto anticipato del processo di addestramento dei modelli in Neptune ML. Tipo: `int`. Default: `3`

  Per informazioni, consulta .

Personalizzazione degli iperparametri in Neptune ML

Quando si modifica il file `model-HPO-configuration.json`, i tipi di modifiche più comuni sono i seguenti:

- Specificare il valore minimo e il valore massimo degli iperparametri `range`.
- Impostare un iperparametro su un valore fisso spostandolo nella sezione `fixed-param` e impostando il relativo valore predefinito sul valore fisso che dovrà assumere.
- Modificare la priorità di un iperparametro posizionandolo in un livello specifico, modificandone l'intervallo e assicurandosi che il valore predefinito sia impostato in modo appropriato.

Best practice per l'addestramento dei modelli

Esistono diverse operazioni consigliate per migliorare le prestazioni dei modelli Neptune ML.

Scelta della proprietà del nodo corretta

Non tutte le proprietà del grafo sono significative o pertinenti per le attività di machine learning. Tutte le proprietà non pertinenti devono essere escluse durante l'esportazione dei dati.

Ecco alcune best practice:

- Rivolgiti a esperti del settore per valutare l'importanza delle funzionalità e la fattibilità del relativo utilizzo per le previsioni.
- Rimuovi le funzionalità che ritieni ridondanti o irrilevanti per ridurre il rumore nei dati e le correlazioni non importanti.
- Esegui iterazioni mentre costruisci il tuo modello. Modifica le funzionalità, le combinazioni di funzionalità e gli obiettivi di ottimizzazione man mano che procedi.

La sezione relativa all'elaborazione delle funzionalità in Amazon Machine Learning Developer Guide fornisce linee guida aggiuntive per l'elaborazione delle funzionalità pertinenti per Neptune ML.

Gestione degli outlier nei punti dati

Un outlier è un punto dati che è significativamente diverso dai dati rimanenti. Gli outlier nei dati possono compromettere o fuorviare il processo di addestramento causando tempi di addestramento più lunghi o modelli meno accurati. A meno che non siano veramente importanti, è consigliabile eliminare gli outlier prima di esportare i dati.

Rimozione di nodi e archi duplicati

Nei grafi archiviati in Neptune possono essere presenti nodi o archi duplicati. Questi elementi ridondanti introdurranno rumore nell'addestramento dei modelli di machine learning. Elimina i nodi o gli archi duplicati prima di esportare i dati.

Ottimizzazione della struttura del grafo

Quando il grafo viene esportato, è possibile modificare la modalità di elaborazione delle funzionalità e la modalità di creazione del grafo, per migliorare le prestazioni del modello.

Ecco alcune best practice:

- Quando una proprietà dell'arco ha lo stesso significato di categorie di archi, in alcuni casi vale la pena trasformarla in tipi di archi.
- La policy di normalizzazione predefinita utilizzata per una proprietà numerica è `min-max`, ma in alcuni casi altre policy di normalizzazione garantiscono un risultato migliore. È possibile pre-elaborare la proprietà e modificare la policy di normalizzazione come spiegato in Elementi di un file `model-HPO-configuration.json`.
- Il processo di esportazione genera automaticamente i tipi di funzionalità in base ai tipi delle proprietà. Ad esempio, tratta le proprietà `String` come funzionalità categoriali e le proprietà `Float` e `Int` come funzionalità numeriche. Se necessario, è possibile modificare il tipo di funzionalità dopo l'esportazione (consulta Elementi di un file `model-HPO-configuration.json`).

Ottimizzazione degli intervalli e dei valori predefiniti degli iperparametri

L'operazione di elaborazione dei dati deduce gli intervalli di configurazione degli iperparametri dal grafo. Se gli intervalli di iperparametri e i valori predefiniti del modello generati non sono adatti ai dati del grafo, puoi modificare il file di configurazione dell'ottimizzazione degli iperparametri per creare una strategia di ottimizzazione degli iperparametri personalizzata.

Ecco alcune best practice:

- Quando il grafo diventa grande, le dimensioni predefinite della dimensione nascosta potrebbero non essere sufficienti per contenere tutte le informazioni. Puoi modificare l'iperparametro `num-hidden` per controllare le dimensioni della dimensione nascosta.
- Per i modelli di incorporamento del grafo della conoscenza, potrebbe essere necessario modificare il modello specifico usato in base alla struttura del grafo e al budget.

  I modelli `TrainsE` hanno difficoltà a gestire le relazioni uno-a-molti (1-N), molti-a-uno (N-1) e molti-a-molti (N-N). I modelli `DistMult` hanno difficoltà a gestire relazioni simmetriche. Il modello `RotatE` consente di modellare tutti i tipi di relazioni ma è più costoso rispetto a `TrainsE` e `DistMult` durante l'addestramento.
- In alcuni casi, quando sono importanti sia l'identificazione dei nodi che le informazioni sulle funzionalità del nodo, è necessario usare `` `concat-node-embed` `` per indicare al modello Neptune ML di recuperare la rappresentazione iniziale di un nodo concatenando le sue funzionalità con gli incorporamenti iniziali.

- Quando si ottengono prestazioni ragionevolmente buone su alcuni iperparametri, è possibile regolare lo spazio di ricerca degli iperparametri in base a tali risultati.

Arresto anticipato del processo di addestramento dei modelli in Neptune ML

L'arresto anticipato può ridurre in modo significativo il tempo di esecuzione dell'addestramento del modello e i costi associati senza compromettere le prestazioni del modello. Inoltre, impedisce l'overfitting del modello sui dati di addestramento.

L'arresto anticipato dipende dalle misurazioni regolari delle prestazioni del set di convalida. Inizialmente, le prestazioni migliorano con il procedere dell'addestramento, ma quando si verifica l'overfitting, peggiora nuovamente. La funzionalità di arresto anticipato identifica il punto di overfitting del modello e arresta l'addestramento del modello in quel momento.

Neptune ML monitora le chiamate alle metriche di convalida e confronta la metrica di convalida più recente con la media delle metriche di convalida delle ultime **n** valutazioni, dove **n** è un numero impostato utilizzando il parametro `window-for-early-stop`. Non appena la metrica di convalida è peggiore di quella media, Neptune ML arresta l'addestramento del modello e salva il miglior modello generato fino a quel momento.

Puoi controllare l'arresto anticipato utilizzando i seguenti parametri:

- **`window-for-early-stop`**: il valore di questo parametro è un numero intero che specifica il numero di punteggi di convalida recenti in base a cui calcolare la media per decidere se effettuare un arresto anticipato. Il valore predefinito è `3`.
- **`enable-early-stop`**: usa questo parametro booleano per disattivare la funzionalità di arresto anticipato. Per impostazione predefinita, il suo valore è `true`.

Arresto anticipato del processo di ottimizzazione degli iperparametri in Neptune ML

La funzione di arresto anticipato di Neptune ML interrompe anche i processi di addestramento che presentano prestazioni peggiori rispetto ad altri processi di aggiornamento, mediante la funzionalità di avvio a caldo dell'ottimizzazione degli iperparametri di SageMaker. Anche questa operazione consente di ridurre i costi e migliorare la qualità dell'ottimizzazione degli iperparametri.

Per una descrizione del funzionamento, consulta Esecuzione di un processo di ottimizzazione degli iperparametri con avvio a caldo.

L'avvio a caldo permette di trasferire le informazioni apprese dai processi di addestramento precedenti ai processi di addestramento successivi e offre due vantaggi distinti:

- Innanzitutto, i risultati dei processi di ottimizzazione precedenti vengono utilizzati per indicare le combinazioni di iperparametri valide da cercare nel nuovo processo di ottimizzazione.
- In secondo luogo, consente l'arresto anticipato per accedere a più esecuzioni del modello, riducendo così i tempi di ottimizzazione.

Questa funzionalità è abilitata automaticamente in Neptune ML e consente di trovare un equilibrio tra il tempo di addestramento del modello e le prestazioni. Se sei soddisfatto delle prestazioni del modello corrente, puoi utilizzare tale modello. In caso contrario, occorre eseguire altre ottimizzazioni degli iperparametri con avvio a caldo con i risultati delle esecuzioni precedenti in modo da individuare un modello migliore.

Servizi di supporto professionali

AWS offre servizi di supporto professionale per aiutarti a risolvere i problemi di machine learning sui progetti Neptune. Se rimani bloccato, contatta il Supporto AWS.

Utilizzo di un modello addestrato per generare nuovi artefatti del modello

Utilizzando il comando di trasformazione dei modelli Neptune ML, puoi calcolare gli artefatti del modello come incorporamenti di nodi sui dati del grafo elaborati usando parametri del modello pre-addestrati.

Trasformazione dei modelli per l'inferenza incrementale

Nel flusso di lavoro di inferenza incrementale del modello, dopo aver esportato da Neptune i dati per l'addestramento, puoi avviare un processo di trasformazione del modello usando un comando curl (o awscurl) come il seguente:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "mlModelTrainingJobId": "(the ML model training job-id)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
      }'
```

Puoi quindi passare l'ID di questo processo alla chiamata API di creazione degli endpoint per creare un nuovo endpoint o aggiornarne uno esistente con i nuovi artefatti del modello generati da questo processo. In questo modo, l'endpoint nuovo o aggiornato potrà fornire previsioni del modello per i dati del grafo aggiornati.

Trasformazione dei modelli per qualsiasi processo di addestramento

Puoi anche specificare un parametro `trainingJobName` per generare gli artefatti del modello per tutti i processi di addestramento SageMaker avviati durante l'addestramento dei modelli Neptune ML. Poiché un processo di addestramento dei modelli Neptune ML può potenzialmente avviare molti processi di addestramento SageMaker, questa opzione ti offre la flessibilità necessaria per creare un endpoint di inferenza basato su uno qualsiasi di questi processi di addestramento SageMaker.

Ad esempio:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "trainingJobName" : "(name a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
      }'
```

Se il processo di addestramento originale riguarda un modello personalizzato fornito dall'utente, è necessario includere un oggetto `customModelTransformParameters` quando si richiama una trasformazione del modello. Per ulteriori informazioni su come implementare e usare un modello personalizzato, consulta Modelli personalizzati in Neptune ML..

> **Note**
>
> Il comando `modeltransform` esegue sempre la trasformazione del modello sul miglior processo di addestramento SageMaker per l'addestramento in questione.

Per ulteriori informazioni sui processi di trasformazione dei modelli, consulta Comando modeltransform.

Artefatti generati dall'addestramento dei modelli in Neptune ML

Dopo l'addestramento del modello, Neptune ML utilizza i parametri del modello con le prestazioni di addestramento migliori per generare gli artefatti del modello necessari per avviare l'endpoint di inferenza e fornire previsioni del modello. Il processo di addestramento crea un pacchetto con gli artefatti e lo archivia nella posizione di output Amazon S3 del miglior processo di addestramento SageMaker.

Le sezioni seguenti descrivono cosa è incluso negli artefatti del modello per le diverse attività e illustrano in che modo il comando di trasformazione del modello usa un modello addestrato preesistente per generare gli artefatti anche su nuovi dati del grafo.

Artefatti generati per diverse attività

Il contenuto degli artefatti del modello generati dal processo di addestramento dipende dall'attività di machine learning di destinazione:

- Classificazione e regressione dei nodi: per la previsione delle proprietà dei nodi, gli artefatti includono i parametri del modello, gli incorporamenti dei nodi dal codificatore GNN, le previsioni del modello per i nodi nel grafo di addestramento e alcuni file di configurazione per l'endpoint di inferenza. Nelle attività di classificazione e regressione dei nodi, le previsioni del modello vengono precalcolate per i nodi presenti durante l'addestramento per ridurre la latenza delle query.
- Classificazione e regressione degli archi: per la previsione delle proprietà degli archi, gli artefatti includono anche i parametri del modello e gli incorporamenti dei nodi. I parametri del decodificatore del modello sono particolarmente importanti per l'inferenza, in quanto la classificazione degli archi o le previsioni di regressione degli archi vengono calcolate applicando il decodificatore del modello agli incorporamenti del vertice di origine e del vertice di destinazione di un arco.
- Previsione dei collegamenti: per la previsione dei collegamenti, oltre agli artefatti generati per la previsione delle proprietà degli archi, anche il grafo DGL viene incluso come artefatto perché la previsione dei collegamenti richiede il grafo di addestramento per eseguire le previsioni. L'obiettivo della previsione dei collegamenti è prevedere i vertici di destinazione che probabilmente si combineranno con un vertice di origine per formare un arco di un tipo specifico nel grafo. A tale scopo, l'incorporamento dei nodi del vertice di origine e una rappresentazione appresa per il tipo di arco vengono combinati con gli incorporamenti dei nodi di tutti i possibili vertici di destinazione per generare un punteggio di probabilità degli archi per ciascuno dei vertici di destinazione. I punteggi vengono quindi ordinati per classificare i potenziali vertici di destinazione e restituire i migliori candidati.

Per ogni tipo di attività, i pesi del modello di rete neurale a grafo (GNN) della libreria DGL vengono salvati nell'artefatto del modello. Questo consente a Neptune ML di calcolare nuovi output del modello man mano che il grafo cambia (inferenza induttiva), oltre a usare previsioni e incorporamenti precalcolati (inferenza trasduttiva) per ridurre la latenza.

Generazione di nuovi artefatti del modello

Gli artefatti del modello generati dopo l'addestramento del modello in Neptune ML sono direttamente collegati al processo di addestramento. Questo significa che gli incorporamenti e le previsioni precalcolati esistono solo per le entità presenti nel grafo di addestramento originale. Sebbene la modalità di inferenza induttiva per gli endpoint Neptune ML possa calcolare previsioni per nuove entità in tempo reale, potrebbe essere necessario generare previsioni in batch su nuove entità senza eseguire query su un endpoint.

Per ottenere previsioni del modello in batch per le nuove entità che sono state aggiunte al grafo, è necessario ricalcolare i nuovi artefatti del modello per i nuovi dati del grafo. Questa operazione viene eseguita utilizzando il comando `modeltransform`. Puoi usare il comando `modeltransform` per ottenere solo previsioni in batch senza configurare un endpoint o per generare tutte le previsioni per poterle riscrivere nel grafo.

Poiché l'addestramento del modello esegue implicitamente una trasformazione del modello alla fine del processo di addestramento, gli artefatti del modello vengono sempre ricalcolati da un processo di addestramento sui dati del grafo di addestramento. Tuttavia, il comando `modeltransform` può anche calcolare gli artefatti del modello su dati del grafi che non sono stati utilizzati per addestrare un modello. A tale scopo, i nuovi dati del grafo devono essere elaborati utilizzando le stesse codifiche di funzionalità dei dati del grafo originale e devono rispettare lo stesso schema del grafo.

Per eseguire questa operazione è prima di tutto necessario creare un nuovo processo di elaborazione dei dati che sia un clone del processo di elaborazione dei dati eseguito sui dati del grafo di addestramento originale ed eseguirlo sui nuovi dati del grafo (consulta Elaborazione dei dati del grafo aggiornati per Neptune ML). Quindi, occorre chiamare il comando `modeltransform` con l'ID `dataProcessingJobId` nuovo e l'ID `modelTrainingJobId` precedente per ricalcolare gli artefatti del modello sui dati del grafo aggiornati.

Per la previsione delle proprietà dei nodi, gli incorporamenti e le previsioni dei nodi vengono ricalcolati sui nuovi dati del grafo, anche per i nodi presenti nel grafo di addestramento originale.

Per la previsione delle proprietà degli archi e la previsione dei collegamenti, anche gli incorporamenti dei nodi vengono ricalcolati e sostituiscono allo stesso modo gli eventuali incorporamenti dei nodi

esistenti. Per ricalcolare gli incorporamenti dei nodi, Neptune ML applica il codificatore GNN appreso dal precedente modello addestrato ai nodi dei nuovi dati del grafo con le nuove funzionalità.

Per i nodi che non dispongono di funzionalità, vengono riutilizzate le rappresentazioni iniziali apprese durante l'addestramento del modello originale. Per i nuovi nodi che non dispongono di funzionalità e che non erano presenti nel grafo di addestramento originale, Neptune ML inizializza la loro rappresentazione come media delle rappresentazioni iniziali apprese dei nodi di quel tipo di nodo presenti nel grafo di addestramento originale. Questo approccio può causare un calo delle prestazioni nelle previsioni del modello se sono presenti molti nuovi nodi privi di funzionalità, poiché verranno tutti inizializzati in base all'incorporamento iniziale medio per quel tipo di nodo.

Se il modello viene addestrato con `concat-node-embed` impostato su true, le rappresentazioni iniziali dei nodi vengono create concatenando le funzionalità dei nodi con la rappresentazione iniziale apprendibile. Pertanto, per il grafo aggiornato, la rappresentazione iniziale dei nuovi nodi utilizza anche gli incorporamenti iniziali medi dei nodi, concatenati con le nuove funzionalità dei nodi.

Inoltre, le eliminazioni dei nodi non sono attualmente supportate. Se i nodi sono stati rimossi nel grafo aggiornato, è necessario riaddestrare il modello in base ai dati del grafo aggiornato.

Il ricalcolo degli artefatti del modello riutilizza i parametri del modello appresi su un nuovo grafo e deve essere eseguito solo quando il nuovo grafo è molto simile a quello precedente. Se il nuovo grafo non è sufficientemente simile, è necessario riaddestrare il modello per ottenere prestazioni simili sui nuovi dati del grafo. Ciò che si intende per sufficientemente simile dipende dalla struttura dei dati del grafo, ma come regola generale è consigliabile riaddestrare il modello se i nuovi dati differiscono di oltre il 10-20% dai dati del grafo di addestramento originale.

Per i grafi in cui tutti i nodi dispongono di funzionalità, si applica il limite superiore della soglia (differenza del 20%), ma per i grafi in cui molti nodi sono privi di funzionalità e i nuovi nodi aggiunti al grafo non hanno proprietà, il limite inferiore (differenza del 10%) potrebbe anche essere troppo alto.

Per ulteriori informazioni sui processi di trasformazione dei modelli, consulta Comando modeltransform.

Modelli personalizzati in Neptune ML.

Neptune ML consente di definire le implementazioni di modelli personalizzati con Python. Puoi addestrare e implementare modelli personalizzati utilizzando l'infrastruttura di Neptune ML come avviene per i modelli predefiniti e utilizzarli per ottenere previsioni tramite query su grafo.

> **Note**
>
> L'inferenza induttiva in tempo reale non è attualmente supportata per i modelli personalizzati.

Puoi iniziare a implementare un modello personalizzato in Python seguendo gli esempi del kit di strumenti per Neptune ML e utilizzando i componenti del modello forniti nel kit di strumenti stesso. Nelle seguenti sezioni sono fornite informazioni dettagliate.

Indice

Panoramica dei modelli personalizzati in Neptune ML

Quando usare un modello personalizzato in Neptune ML

I modelli predefiniti di Neptune ML gestiscono tutte le attività standard supportate da Neptune ML, ma in alcuni casi potrebbe essere necessario un controllo più granulare sul modello per un'attività specifica o personalizzare il processo di addestramento del modello. Un modello personalizzato, ad esempio, è appropriato nelle seguenti situazioni:

- È necessario eseguire la codifica delle funzionalità per le funzionalità di testo di modelli di testo molto grandi su istanze GPU.
- Vuoi usare un modello di rete neurale a grafo (GNN) personalizzato sviluppato in Deep Graph Library (DGL).
- Vuoi usare modelli tabulari o modelli di insieme per la classificazione e la regressione dei nodi.

Flusso di lavoro per lo sviluppo e l'utilizzo di un modello personalizzato in Neptune ML

Il supporto di modelli personalizzati in Neptune ML è progettato per integrarsi perfettamente nei flussi di lavoro Neptune ML esistenti. Il codice personalizzato viene eseguito nel modulo di origine sull'infrastruttura di Neptune ML per addestrare il modello. Proprio come nel caso di un modello predefinito, Neptune ML avvia automaticamente un processo di ottimizzazione degli iperparametri SageMaker e seleziona il modello migliore in base alla metrica di valutazione. Utilizza quindi l'implementazione fornita nel modulo di origine per generare gli artefatti del modello da implementare.

Le fasi di esportazione dei dati, configurazione di addestramento e pre-elaborazione dei dati sono le stesse sia per i modelli personalizzati che per quelli predefiniti.

Dopo la pre-elaborazione dei dati è possibile sviluppare e testare l'implementazione del modello personalizzato in modo iterativo e interattivo con Python. Quando il modello è pronto per la produzione, puoi caricare il modulo Python risultante su Amazon S3 in questo modo:

```
aws s3 cp --recursive (source path to module) s3://(bucket name)/(destination path for
 your module)
```

Quindi, puoi utilizzare il normale flusso di lavoro dei dati predefinito o incrementale per implementare il modello in produzione, con alcune differenze.

Per l'addestramento dei modelli con un modello personalizzato, è necessario fornire un oggetto JSON `customModelTrainingParameters` all'API di addestramento dei modelli

Neptune ML per assicurarsi che venga usato il codice personalizzato. I campi nell'oggetto `customModelTrainingParameters` sono i seguenti:

- **`sourceS3DirectoryPath`** (obbligatorio): percorso della posizione Amazon S3 in cui si trova il modulo Python che implementa il modello. Deve indicare una posizione Amazon S3 esistente valida che contenga almeno uno script di addestramento, uno script di trasformazione e un file `model-hpo-configuration.json`.
- **`trainingEntryPointScript`** (facoltativo): nome del punto di ingresso nel modulo di uno script che esegue l'addestramento del modello e accetta gli iperparametri come argomenti della riga di comando, inclusi gli iperparametri fissi.

  Default: `training.py`
- **`transformEntryPointScript`** (facoltativo): nome del punto di ingresso nel modulo di uno script che deve essere eseguito dopo aver identificato il modello migliore ottenuto dalla ricerca degli iperparametri, in modo da calcolare gli artefatti del modello necessari per l'implementazione del modello. Deve poter essere eseguito senza argomenti della riga di comando.

  Default: `transform.py`

Ad esempio:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Allo stesso modo, per abilitare una trasformazione del modello personalizzata, è necessario fornire un oggetto JSON `customModelTransformParameters` all'API di trasformazione dei modelli Neptune ML, con valori dei campi compatibili con i parametri del modello salvati dal processo di addestramento. L'oggetto `customModelTransformParameters` contiene i seguenti campi:

- **`sourceS3DirectoryPath`** (obbligatorio): percorso della posizione Amazon S3 in cui si trova il modulo Python che implementa il modello. Deve indicare una posizione Amazon S3 esistente valida che contenga almeno uno script di addestramento, uno script di trasformazione e un file `model-hpo-configuration.json`.
- **`transformEntryPointScript`** (facoltativo): nome del punto di ingresso nel modulo di uno script che deve essere eseguito dopo aver identificato il modello migliore ottenuto dalla ricerca degli iperparametri, in modo da calcolare gli artefatti del modello necessari per l'implementazione del modello. Deve poter essere eseguito senza argomenti della riga di comando.

  Default: `transform.py`

Ad esempio:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
        "customModelTransformParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Sviluppo di modelli personalizzati in Neptune ML

Un punto di partenza valido per lo sviluppo di modelli personalizzati è seguire gli esempi del kit di strumenti per Neptune ML per strutturare e scrivere il modulo di addestramento. Il kit di strumenti per Neptune ML implementa anche componenti modularizzati del modello ML a grafo nel modelzoo che puoi combinare e usare per creare un modello personalizzato.

Inoltre, il kit di strumenti offre funzioni di utilità che consentono di generare gli artefatti necessari durante l'addestramento e la trasformazione dei modelli. Puoi importare questo pacchetto Python nella tua implementazione personalizzata. Tutte le funzioni o i moduli forniti nel kit di strumenti sono disponibili anche nell'ambiente di addestramento Neptune ML.

Se il tuo modulo Python contiene dipendenze esterne aggiuntive, puoi includere queste dipendenze aggiuntive creando un file `requirements.txt` nella directory del modulo. I pacchetti elencati nel file `requirements.txt` verranno quindi installati prima dell'esecuzione dello script di addestramento.

Come minimo, il modulo Python che implementa il modello personalizzato deve contenere quanto segue:

- Un punto di ingresso dello script di addestramento
- Un punto di ingresso dello script di trasformazione
- Un file `model-hpo-configuration.json`

Sviluppo di script di addestramento dei modelli personalizzati in Neptune ML

Lo script di addestramento del modello personalizzato deve essere uno script Python eseguibile come l'esempio `train.py` del kit di strumenti per Neptune ML. Deve accettare nomi e valori di iperparametri come argomenti della riga di comando. Durante l'addestramento del modello, i nomi degli iperparametri vengono ottenuti dal file `model-hpo-configuration.json`. I valori degli iperparametri rientrano nell'intervallo di iperparametri validi se l'iperparametro è ottimizzabile oppure assumono il valore di iperparametro predefinito se non è ottimizzabile.

Lo script di addestramento viene eseguito su un'istanza di addestramento SageMaker utilizzando una sintassi come questa:

```
python3 (script entry point) --(1st parameter) (1st value) --(2nd parameter) (2nd
 value) (...)
```

Per tutte le attività, Neptune ML AutoTrainer invia numerosi parametri obbligatori allo script di addestramento oltre agli iperparametri specificati e lo script deve poter gestire questi parametri aggiuntivi per funzionare correttamente.

Questi parametri aggiuntivi obbligatori variano leggermente in base all'attività:

Per la classificazione o la regressione dei nodi

- **`task`**: tipo di attività utilizzato internamente da Neptune ML. Per la classificazione dei nodi è `node_class` e per la regressione dei nodi è `node_regression`.
- **`model`**: nome del modello utilizzato internamente da Neptune ML, che in questo caso è `custom`.
- **`name`**: nome dell'attività utilizzata internamente da Neptune ML, che in questo caso è `node_class-custom` per la classificazione dei nodi e `node_regression-custom` per la regressione dei nodi.
- **`target_ntype`**: nome del tipo di nodo per la classificazione o la regressione.
- **`property`**: nome della proprietà del nodo per la classificazione o la regressione.

Per la previsione dei collegamenti

- **`task`**: tipo di attività utilizzato internamente da Neptune ML. Per la previsione dei collegamenti è `link_predict`.
- **`model`**: nome del modello utilizzato internamente da Neptune ML, che in questo caso è `custom`.
- **`name`**: nome dell'attività utilizzata internamente da Neptune ML, che in questo caso è `link_predict-custom`.

Per la classificazione o la regressione degli archi

- **`task`**: tipo di attività utilizzato internamente da Neptune ML. Per la classificazione degli archi è `edge_class` e per la regressione degli archi è `edge_regression`.
- **`model`**: nome del modello utilizzato internamente da Neptune ML, che in questo caso è `custom`.
- **`name`**: nome dell'attività utilizzata internamente da Neptune ML, che in questo caso è `edge_class-custom` per la classificazione degli archi e `edge_regression-custom` per la regressione degli archi.
- **`target_etype`**: nome del tipo di arco per la classificazione o la regressione.
- **`property`**: nome della proprietà dell'arco per la classificazione o la regressione.

Lo script deve salvare i parametri del modello e tutti gli altri artefatti necessari al termine dell'addestramento.

È possibile utilizzare le funzioni di utilità del kit di strumenti per Neptune ML per determinare la posizione dei dati del grafo elaborati, la posizione in cui devono essere salvati i parametri del modello e quali dispositivi GPU sono disponibili sull'istanza di addestramento. Per esempi di utilizzo di queste funzioni di utilità, consulta lo script di addestramento di esempio train.py.

Sviluppo di script di trasformazione dei modelli personalizzati in Neptune ML

Uno script di trasformazione consente di sfruttare il flusso di lavoro incrementale di Neptune ML per l'inferenza del modello su grafi in evoluzione senza riaddestrare il modello. Anche se tutti gli artefatti necessari per l'implementazione del modello vengono generati dallo script di addestramento, è comunque necessario fornire uno script di trasformazione se si desidera generare modelli aggiornati senza riaddestrare il modello.

> **Note**
>
> L'inferenza induttiva in tempo reale non è attualmente supportata per i modelli personalizzati.

Lo script di trasformazione del modello personalizzato deve essere uno script Python eseguibile come lo script di esempio transform.py del kit di strumenti per Neptune ML. Poiché questo script viene richiamato durante l'addestramento del modello senza argomenti della riga di comando, tutti gli argomenti della riga di comando che lo script accetta devono avere valori predefiniti.

Lo script viene eseguito su un'istanza di addestramento SageMaker con una sintassi come questa:

```
python3 (your transform script entry point)
```

Lo script di trasformazione necessita di numerose informazioni, ad esempio:

- Posizione dei dati del grafo elaborati.
- Posizione in cui vengono salvati i parametri del modello e in cui devono essere salvati i nuovi artefatti del modello.
- Dispositivi disponibili sull'istanza.
- Iperparametri che hanno generato il modello migliore.

Questi input vengono ottenuti utilizzando le funzioni di utilità Neptune ML che lo script può chiamare. Consulta lo script transform.py di esempio del kit di strumenti per esempi su come procedere.

Lo script deve salvare gli incorporamenti dei nodi, le mappature degli ID dei nodi e gli eventuali altri artefatti necessari per l'implementazione del modello per ogni attività. Consulta la documentazione degli artefatti del modello per ulteriori informazioni sugli artefatti del modello necessari per le diverse attività di Neptune ML.

File **`model-hpo-configuration.json`** personalizzato in Neptune ML

Il file `model-hpo-configuration.json` definisce gli iperparametri per il modello personalizzato. È nello stesso formato del file `model-hpo-configuration.json` usato con i modelli predefiniti di Neptune ML e ha la precedenza sulla versione generata automaticamente da Neptune ML e caricata nella posizione dei dati elaborati.

Quando aggiungi un nuovo iperparametro al modello, devi anche aggiungere una voce per l'iperparametro in questo file in modo che l'iperparametro venga passato allo script di addestramento.

È necessario fornire un intervallo per l'iperparametro se vuoi che sia ottimizzabile e devi impostarlo come parametro `tier-1`, `tier-2` o `tier-3`. L'iperparametro verrà ottimizzato se il numero totale di processi di addestramento configurati consente di ottimizzare gli iperparametri nel relativo livello. Per un parametro non ottimizzabile, è necessario fornire un valore predefinito e aggiungere l'iperparametro alla sezione `fixed-param` del file. Per un esempio di come eseguire questa operazione, consulta il file `model-hpo-configuration.json` di esempio del kit di strumenti.

È inoltre necessario fornire la definizione della metrica che il processo di ottimizzazione degli iperparametri SageMaker userà per valutare i modelli candidati addestrati. A tale scopo, aggiungi un oggetto JSON `eval_metric` al file `model-hpo-configuration.json` in questo modo:

```
"eval_metric": {
  "tuning_objective": {
      "MetricName": "(metric_name)",
      "Type": "Maximize"
  },
  "metric_definitions": [
    {
      "Name": "(metric_name)",
      "Regex": "(metric regular expression)"
    }
  ]
```

```
},
```

L'array `metric_definitions` nell'oggetto `eval_metric` elenca gli oggetti definizione della metrica per ogni metrica che SageMaker dovrà estrarre dall'istanza di addestramento. Ogni oggetto definizione della metrica ha una chiave `Name` che consente di specificare un nome per la metrica, ad esempio "accuratezza", "F1" e così via. La chiave `Regex` permette di fornire una stringa di espressione regolare che corrisponde al modo in cui la metrica specifica viene stampata nei log di addestramento. Per informazioni dettagliate sulla definizione delle metriche, consulta la pagina Ottimizzazione degli iperparametri SageMaker.

L'oggetto `tuning_objective` in `eval_metric` consente quindi di specificare quale metrica tra quelle in `metric_definitions` dovrà essere usata come metrica di valutazione che funge da metrica obiettivo per l'ottimizzazione degli iperparametri. Il valore di `MetricName` deve corrispondere al valore di `Name` a in una delle definizioni in `metric_definitions`. Il valore di `Type` deve essere "Maximize" o "Minimize" a seconda che la metrica debba essere interpretata come maggiore è migliore (ad esempio "accuratezza") o minore è migliore (ad esempio "errore quadratico medio").

Gli errori in questa sezione del file `model-hpo-configuration.json` possono causare errori nel processo dell'API di addestramento dei modelli Neptune ML, perché il processo di ottimizzazione degli iperparametri SageMaker non sarà in grado di selezionare il modello migliore.

Test locale dell'implementazione del modello personalizzato in Neptune ML

Puoi usare l'ambiente Conda del kit di strumenti per Neptune ML per eseguire il codice in locale per il test e la convalida del modello. Se stai sviluppando in un'istanza notebook Neptune, l'ambiente Conda sarà preinstallato nell'istanza notebook Neptune. Se stai sviluppando in un'istanza diversa, devi seguire le istruzioni di configurazione per l'ambiente locale nel kit di strumenti per Neptune ML.

L'ambiente Conda riproduce accuratamente l'ambiente in cui verrà eseguito il modello quando chiami l'API di addestramento dei modelli. Tutti gli script di addestramento e gli script di trasformazione di esempio consentono di passare un flag `--local` della riga di comando per eseguire gli script in un ambiente locale per facilitare il debug. Si tratta di una procedura valida durante lo sviluppo di un modello personalizzato perché consente di testare l'implementazione del modello in modo interattivo e iterativo. Durante l'addestramento del modello nell'ambiente di addestramento di produzione Neptune ML, questo parametro viene omesso.

Creazione di un endpoint di inferenza per l'esecuzione di query

Un endpoint di inferenza consente di eseguire query su un modello specifico creato dal processo di addestramento del modello. L'endpoint si collega al modello con le migliori prestazioni di un tipo specifico che il processo di addestramento è riuscito a generare. L'endpoint può quindi accettare le query Gremlin di Neptune e restituire le previsioni di quel modello per gli input nelle query. Dopo aver creato l'endpoint di inferenza, questo rimane attivo finché non viene eliminato.

Gestione degli endpoint di inferenza per Neptune ML

Dopo aver completato l'addestramento del modello sui dati esportati da Neptune, puoi creare un endpoint di inferenza usando un comando `curl` (o `awscurl`) come il seguente:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Puoi anche creare un endpoint di inferenza da un modello creato da un processo di trasformazione del modello completato, più o meno allo stesso modo:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTransformJobId": "(the model-transform job-id of a completed job)"
      }'
```

I dettagli su come usare questi comandi sono spiegati in Comando endpoints, insieme alle informazioni su come recuperare lo stato di un endpoint, come eliminare un endpoint e come elencare tutti gli endpoint di inferenza.

Query di inferenza in Neptune ML

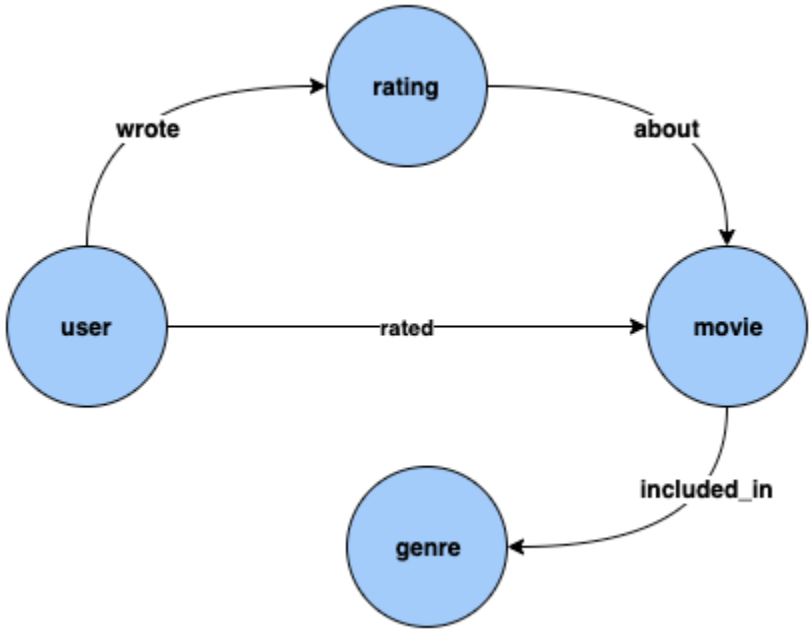
Per eseguire query sull'endpoint di inferenza Neptune ML è possibile usare Gremlin o SPARQL. L'inferenza induttiva in tempo reale, tuttavia, è attualmente supportata solo per le query Gremlin.

Query di inferenza Gremlin in Neptune ML

Come descritto in Funzionalità di Neptune ML, Neptune ML supporta modelli di addestramento che possono eseguire i seguenti tipi di attività di inferenza:

- Classificazione dei nodi: previsione della funzionalità categoriale di una proprietà del vertice.
- Regressione dei nodi: previsione di una proprietà numerica di un vertice.
- Classificazione degli archi: previsione della funzionalità categoriale di una proprietà dell'arco.
- Regressione degli archi: previsione di una proprietà numerica di un arco.
- Previsione dei collegamenti: previsione dei nodi di destinazione per un nodo di origine e un arco in uscita specifici oppure i nodi di origine per un nodo di destinazione e un arco in entrata specificati.

Queste attività verranno illustrate con esempi che utilizzano il set di dati MovieLens 100k fornito da GroupLens Research. Questo set di dati è costituito da film, utenti e valutazioni dei film da parte degli utenti, da cui è stato creato un grafo delle proprietà come questo:



Classificazione dei nodi: nel set di dati precedente, `Genre` è un tipo di vertice collegato al tipo di vertice `Movie` tramite l'arco `included_in`. Tuttavia, se si modifica il set di dati per rendere `Genre` una funzionalità categoriale per il tipo di vertice `Movie`, il problema di dedurre `Genre` per i nuovi film aggiunti al grafo della conoscenza può essere risolto usando modelli di classificazione dei nodi.

Regressione dei nodi: se si considera il tipo di vertice `Rating`, che ha proprietà come `timestamp` e `score`, il problema di dedurre il valore numerico `Score` per una valutazione `Rating` può essere risolto usando modelli di regressione dei nodi.

Classificazione degli archi: allo stesso modo, se per un arco `Rated` è presente una proprietà `Scale` che può avere uno dei valori `Love`, `Like`, `Dislike`, `Neutral`, `Hate`, allora il problema di dedurre `Scale` per l'arco `Rated` per i nuovi film/valutazioni può essere risolto utilizzando modelli di classificazione degli archi.

Regressione degli archi: allo stesso modo, se per lo stesso arco `Rated` è presente una proprietà `Score` che contiene un valore numerico per la valutazione, questo può essere dedotto usando modelli di regressione degli archi.

Previsione dei collegamenti: nell'ambito della previsione dei collegamenti rientrano problemi come trovare i dieci utenti con le maggiori probabilità di valutare un determinato film o trovare i primi dieci film con le maggiori probabilità di essere valutati da un utente specifico.

> **Note**
>
> Per i casi d'uso di Neptune ML, è disponibile un ampio set di notebook progettati per fornire informazioni pratiche su ogni caso d'uso. È possibile creare questi notebook insieme al cluster Neptune quando si utilizza il modello AWS CloudFormation Neptune ML per creare un cluster Neptune ML. Questi notebook sono disponibili anche su github.

Argomenti

Predicati Neptune ML utilizzati nelle query di inferenza Gremlin

`Neptune#ml.deterministic`

Questo predicato è un'opzione per le query di inferenza induttiva, ovvero per le query che includono il predicato `Neptune#ml.inductiveInference`.

Quando si utilizza l'inferenza induttiva, il motore Neptune crea il sottografo appropriato per valutare il modello GNN addestrato e i requisiti di questo sottografo dipendono dai parametri del modello finale. In particolare, il parametro `num-layer` determina il numero di hop di attraversamento dai nodi o dagli archi di destinazione e il parametro `fanouts` specifica il numero di vicini da campionare in ogni hop (consulta Parametri di ottimizzazione degli iperparametri).

Per impostazione predefinita, le query di inferenza induttiva vengono eseguite in modalità non deterministica, in cui Neptune crea il neighborhood in modo casuale. Quando si effettuano previsioni, questo normale campionamento casuale dei vicini genera talvolta previsioni diverse.

Quando si include `Neptune#ml.deterministic` in una query di inferenza induttiva, il motore Neptune tenta di campionare i vicini in modo deterministico in modo che più invocazioni della stessa query restituiscano sempre gli stessi risultati. Tuttavia, non si può garantire che i risultati siano completamente deterministici, poiché le modifiche al grafo e agli artefatti sottostanti dei sistemi distribuiti possono comunque introdurre fluttuazioni.

Includi il predicato `Neptune#ml.deterministic` in una query in questo modo:

```
.with("Neptune#ml.deterministic")
```

Se il predicato `Neptune#ml.deterministic` è incluso in una query che non include anche `Neptune#ml.inductiveInference`, viene semplicemente ignorato.

`Neptune#ml.disableInductiveInferenceMetadataCache`

Questo predicato è un'opzione per le query di inferenza induttiva, ovvero per le query che includono il predicato `Neptune#ml.inductiveInference`.

Per le query di inferenza induttiva, Neptune utilizza un file di metadati archiviato in Amazon S3 per decidere il numero di hop e il fanout durante la creazione del neighborhood Neptune normalmente memorizza nella cache i metadati di questo modello per evitare di recuperare ripetutamente il file da Amazon S3. La memorizzazione nella cache può essere disabilitata includendo il predicato `Neptune#ml.disableInductiveInferenceMetadataCache` nella query. Sebbene per Neptune l'operazione di recupero dei metadati direttamente da Amazon S3 sia più lenta, è utile quando

l'endpoint SageMaker è stato aggiornato dopo il riaddestramento o la trasformazione e la cache non è aggiornata.

Includi il predicato `Neptune#ml.disableInductiveInferenceMetadataCache` in una query in questo modo:

```
.with("Neptune#ml.disableInductiveInferenceMetadataCache")
```

Ecco l'aspetto di una query di esempio in un notebook Jupyter:

```
%%gremlin
g.with("Neptune#ml.endpoint", "ep1")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .with("Neptune#ml.disableInductiveInferenceMetadataCache")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
```

`Neptune#ml.endpoint`

Il predicato `Neptune#ml.endpoint` viene utilizzato in un passaggio `with()` per specificare l'endpoint di inferenza, se necessario:

```
 .with("Neptune#ml.endpoint", "the model's SageMaker inference endpoint")
```

È possibile identificare l'endpoint in base all'`id` o al relativo URL. Ad esempio:

```
 .with( "Neptune#ml.endpoint", "node-classification-movie-lens-endpoint" )
```

O:

```
 .with( "Neptune#ml.endpoint", "https://runtime.sagemaker.us-east-1.amazonaws.com/
endpoints/node-classification-movie-lens-endpoint/invocations" )
```

> **Note**
>
> Se imposti il parametro `neptune_ml_endpoint` nel gruppo di parametri del cluster database Neptune DB sull'`id` o sull'URL dell'endpoint, non è necessario includere il predicato `Neptune#ml.endpoint` in ogni query.

Neptune#ml.iamRoleArn

`Neptune#ml.iamRoleArn` viene usato in un passaggio `with()` per specificare il nome ARN del ruolo IAM dell'esecuzione SageMaker, se necessario:

```
.with("Neptune#ml.iamRoleArn", "the ARN for the SageMaker execution IAM role")
```

Per informazioni su come creare il ruolo IAM dell'esecuzione SageMaker, consulta Per creare un ruolo `NeptuneSageMakerIAMRole` personalizzato.

> **Note**
>
> Se imposti il parametro `neptune_ml_iam_role` nel gruppo di parametri del cluster database Neptune DB sul nome ARN del ruolo IAM dell'esecuzione SageMaker, non è necessario includere il predicato `Neptune#ml.iamRoleArn` in ogni query.

Neptune#ml.inductiveInference

In Gremlin l'inferenza trasduttiva è abilitata per impostazione predefinita. Per creare una query di inferenza induttiva in tempo reale, includi il predicato `Neptune#ml.inductiveInference` in questo modo:

```
.with("Neptune#ml.inductiveInference")
```

Se il grafo è dinamico, l'inferenza induttiva è spesso la scelta migliore, ma se il grafo è statico, l'inferenza trasduttiva è più veloce ed efficiente.

Neptune#ml.limit

Il predicato `Neptune#ml.limit` limita facoltativamente il numero di risultati restituiti per ogni entità:

```
.with( "Neptune#ml.limit", 2 )
```

Per impostazione predefinita, il limite è 1 e il numero massimo che è possibile impostare è 100.

Neptune#ml.threshold

Il predicato `Neptune#ml.threshold` stabilisce facoltativamente una soglia limite per i punteggi dei risultati:

```
.with( "Neptune#ml.threshold", 0.5D )
```

Ciò consente di scartare tutti i risultati con punteggi inferiori alla soglia specificata.

Neptune#ml.classification

Il predicato `Neptune#ml.classification` è collegato al passaggio `properties()` per stabilire che è necessario recuperare le proprietà dall'endpoint SageMaker del modello di classificazione dei nodi:

```
.properties( "property key of the node classification
model" ).with( "Neptune#ml.classification" )
```

Neptune#ml.regression

Il predicato `Neptune#ml.regression` è collegato al passaggio `properties()` per stabilire che è necessario recuperare le proprietà dall'endpoint SageMaker del modello di regressione dei nodi:

```
.properties( "property key of the node regression
model" ).with( "Neptune#ml.regression" )
```

Neptune#ml.prediction

Il predicato `Neptune#ml.prediction` è collegato ai passaggi `in()` e `out()` per stabilire che si tratta di una query di previsione dei collegamenti:

```
.in("edge label of the link prediction
model").with("Neptune#ml.prediction").hasLabel("target node label")
```

Neptune#ml.score

Il predicato `Neptune#ml.score` viene usato nelle query di classificazione dei nodi o degli archi Gremlin per ottenere un punteggio di attendibilità di machine learning. Il predicato `Neptune#ml.score` deve essere passato insieme al predicato di query nel passaggio `properties()` per ottenere un punteggio di attendibilità di ML per le query di classificazione dei nodi o degli archi.

È possibile trovare un esempio di classificazione dei nodi con altri esempi di classificazione dei nodi e un esempio di classificazione degli archi nella sezione relativa alla classificazione degli archi.

Query di classificazione dei nodi Gremlin in Neptune ML

Per la classificazione dei nodi Gremlin in Neptune ML

- Il modello viene addestrato su una proprietà dei vertici. Il set di valori univoci di questa proprietà è definito come un set di classi del nodo o semplicemente classi.
- La classe del nodo o il valore della proprietà categoriale di un vertice può essere dedotto dal modello di classificazione dei nodi. Questo è utile quando questa proprietà non è già collegata al vertice.
- Per recuperare una o più classi da un modello di classificazione dei nodi, è necessario utilizzare il passaggio `with()` con il predicato `Neptune#ml.classification` per configurare il passaggio `properties()`. Il formato di output è simile a quello previsto se si trattasse di proprietà dei vertici.

> **Note**
>
> La classificazione dei nodi funziona solo con i valori delle proprietà stringa. Ciò significa che i valori delle proprietà numeriche come `0` o `1` non sono supportati, sebbene gli equivalenti stringa `"0"` e `"1"` lo siano. Allo stesso modo, i valori delle proprietà booleane `true` e `false` non sono supportati, ma `"true"` e `"false"` funzionano.

Ecco una query di classificazione dei nodi di esempio:

```
g.with( "Neptune#ml.endpoint","node-classification-movie-lens-endpoint" )
 .with( "Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role" )
 .with( "Neptune#ml.limit", 2 )
 .with( "Neptune#ml.threshold", 0.5D )
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre").with("Neptune#ml.classification")
```

L'output di questa query ha un aspetto simile al seguente:

```
==>vp[genre->Action]
==>vp[genre->Crime]
==>vp[genre->Comedy]
```

Nella query precedente i passaggi `properties()` e `V()` vengono utilizzati nel modo seguente:

Il passaggio `V()` contiene il set di vertici per cui recuperare le classi dal modello di classificazione dei nodi:

```
  .V( "movie_1", "movie_2", "movie_3" )
```

Il passaggio `properties()` contiene la chiave su cui è stato addestrato il modello e include `.with("Neptune#ml.classification")` per indicare che si tratta di una query di inferenza ML per la classificazione dei nodi.

Attualmente non sono supportate più chiavi delle proprietà in un passaggio `properties().with("Neptune#ml.classification")`. Ad esempio, la query seguente restituisce un'eccezione:

```
g.with("Neptune#ml.endpoint", "node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre", "other_label").with("Neptune#ml.classification")
```

Per il messaggio di errore specifico, consulta l'elenco delle eccezioni di Neptune ML.

È possibile usare un passaggio `properties().with("Neptune#ml.classification")` in combinazione con uno dei seguenti passaggi:

- `value()`
- `value().is()`
- `hasValue()`
- `has(value,"")`
- `key()`
- `key().is()`
- `hasKey()`
- `has(key,"")`
- `path()`

Altre query di classificazione dei nodi

Se sia l'endpoint di inferenza che il ruolo IAM corrispondente sono stati salvati nel gruppo di parametri del cluster database, una query di classificazione dei nodi può essere semplice come questa:

```
g.V("movie_1", "movie_2",
 "movie_3").properties("genre").with("Neptune#ml.classification")
```

È possibile combinare proprietà e classi dei vertici in una query usando il passaggio `union()`:

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .union(
   properties("genre").with("Neptune#ml.classification"),
   properties("genre")
 )
```

Puoi anche creare una query illimitata come questa:

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V()
 .properties("genre").with("Neptune#ml.classification")
```

Puoi recuperare le classi dei nodi insieme ai vertici usando il passaggio `select()` insieme al passaggio `as()`:

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" ).as("vertex")
 .properties("genre").with("Neptune#ml.classification").as("properties")
 .select("vertex","properties")
```

Puoi anche filtrare in base alle classi di nodi, come illustrato in questi esempi:

```
g.with("Neptune#ml.endpoint", "node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
```

```
  .properties("genre").with("Neptune#ml.classification")
  .has(value, "Horror")

g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V( "movie_1", "movie_2", "movie_3" )
  .properties("genre").with("Neptune#ml.classification")
  .has(value, P.eq("Action"))

g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V( "movie_1", "movie_2", "movie_3" )
  .properties("genre").with("Neptune#ml.classification")
  .has(value, P.within("Action", "Horror"))
```

È possibile ottenere un punteggio di attendibilità di classificazione dei nodi usando il predicato `Neptune#ml.score`:

```
 g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre", "Neptune#ml.score").with("Neptune#ml.classification")
```

E la risposta avrà un aspetto simile al seguente:

```
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.01234567]
==>vp[genre->Crime]
==>vp[Neptune#ml.score->0.543210]
==>vp[genre->Comedy]
==>vp[Neptune#ml.score->0.10101]
```

Utilizzo dell'inferenza induttiva in una query di classificazione dei nodi

Si supponga di dover aggiungere un nuovo nodo a un grafo esistente in un notebook Jupyter, in questo modo:

```
%%gremlin
g.addV('label1').property(id,'101').as('newV')
 .V('1').as('oldV1')
 .V('2').as('oldV2')
 .addE('eLabel1').from('newV').to('oldV1')
```

```
  .addE('eLabel2').from('oldV2').to('newV')
```

È quindi possibile usare una query di inferenza induttiva per ottenere un genere e un punteggio di attendibilità che riflettano il nuovo nodo:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("genre", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
```

Se la query è stata eseguita più volte, tuttavia, i risultati restituiti potrebbero essere leggermente diversi:

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.21365921]
```

Puoi decidere rendere deterministica la stessa query:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("genre", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

In tal caso, i risultati sarebbero più o meno gli stessi ogni volta:

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
```

Query di regressione dei nodi Gremlin in Neptune ML

La regressione dei nodi è simile alla classificazione dei nodi, tranne per il fatto che il valore dedotto dal modello di regressione per ogni nodo è numerico. Per la regressione dei nodi è possibile usare le stesse query Gremlin usate per la classificazione dei nodi, con le seguenti differenze:

- Ancora una volta, in Neptune ML i nodi fanno riferimento ai vertici.
- Il passaggio `properties()` accetta il formato `properties().with("Neptune#ml.regression")` anziché `properties().with("Neptune#ml.classification")`.
- I predicati `"Neptune#ml.limit"` e `"Neptune#ml.threshold"` non sono applicabili.
- Quando si filtra in base al valore, è necessario specificare un valore numerico.

Ecco una query di classificazione dei vertici di esempio:

```
g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1","movie_2","movie_3")
 .properties("revenue").with("Neptune#ml.regression")
```

È possibile filtrare in base al valore dedotto utilizzando un modello di regressione, come illustrato negli esempi seguenti:

```
g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1","movie_2","movie_3")
 .properties("revenue").with("Neptune#ml.regression")
 .value().is(P.gte(1600000))

g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1","movie_2","movie_3")
 .properties("revenue").with("Neptune#ml.regression")
 .hasValue(P.lte(1600000D))
```

Utilizzo dell'inferenza induttiva in una query di regressione dei nodi

Si supponga di dover aggiungere un nuovo nodo a un grafo esistente in un notebook Jupyter, in questo modo:

```
%%gremlin
g.addV('label1').property(id,'101').as('newV')
 .V('1').as('oldV1')
 .V('2').as('oldV2')
 .addE('eLabel1').from('newV').to('oldV1')
 .addE('eLabel2').from('oldV2').to('newV')
```

È quindi possibile utilizzare una query di inferenza induttiva per ottenere una valutazione che tenga conto del nuovo nodo:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nr-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
```

Poiché la query è non deterministica, potrebbe restituire risultati leggermente diversi se la si esegue più volte, in base al neighborhood:

```
# First time
==>vp[rating->9.1]

# Second time
==>vp[rating->8.9]
```

Se sono necessari risultati più coerenti, puoi rendere la query deterministica:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Ora i risultati saranno più o meno gli stessi ogni volta:

```
# First time
==>vp[rating->9.1]
```

```
# Second time
==>vp[rating->9.1]
```

Query di classificazione degli archi Gremlin in Neptune ML

Per la classificazione degli archi Gremlin in Neptune ML:

- Il modello viene addestrato su una proprietà degli archi. Il set di valori univoci di questa proprietà è definito come un set di classi.
- Il valore della classe o della proprietà categoriale di un arco può essere dedotto dal modello di classificazione degli archi, utile quando questa proprietà non è già collegata all'arco.
- Per recuperare una o più classi da un modello di classificazione degli archi, è necessario utilizzare il passaggio `with()` con il predicato `"Neptune#ml.classification"` per configurare il passaggio `properties()`. Il formato di output è simile a quello previsto se si trattasse di proprietà degli archi.

> **Note**
>
> La classificazione degli archi funziona solo con i valori delle proprietà stringa. Ciò significa che i valori delle proprietà numeriche come `0` o `1` non sono supportati, sebbene gli equivalenti stringa `"0"` e `"1"` lo siano. Allo stesso modo, i valori delle proprietà booleane `true` e `false` non sono supportati, ma `"true"` e `"false"` funzionano.

Ecco un esempio di query di classificazione degli archi che richiede un punteggio di attendibilità usando il predicato `Neptune#ml.score` :

```
g.with("Neptune#ml.endpoint","edge-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by", "Neptune#ml.score").with("Neptune#ml.classification")
```

E la risposta avrà un aspetto simile al seguente:

```
==>p[knows_by->"Family"]
==>p[Neptune#ml.score->0.01234567]
==>p[knows_by->"Friends"]
```

```
==>p[Neptune#ml.score->0.543210]
==>p[knows_by->"Colleagues"]
==>p[Neptune#ml.score->0.10101]
```

Sintassi di una query di classificazione degli archi Gremlin

Per un grafo semplice in cui `User` è il nodo di testa e il nodo di coda e `Relationship` è l'arco che li collega, ecco un esempio di query di classificazione degli archi:

```
g.with("Neptune#ml.endpoint","edge-classification-social-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by").with("Neptune#ml.classification")
```

L'output di questa query ha un aspetto simile al seguente:

```
==>p[knows_by->"Family"]
==>p[knows_by->"Friends"]
==>p[knows_by->"Colleagues"]
```

Nella query precedente i passaggi `properties()` e `E()` vengono utilizzati nel modo seguente:

- Il passaggio `E()` contiene il set di archi per cui recuperare le classi dal modello di classificazione degli archi:

  ```
  .E("relationship_1","relationship_2","relationship_3")
  ```

- Il passaggio `properties()` contiene la chiave su cui è stato addestrato il modello e include `.with("Neptune#ml.classification")` per indicare che si tratta di una query di inferenza ML per la classificazione degli archi.

Attualmente non sono supportate più chiavi delle proprietà in un passaggio `properties().with("Neptune#ml.classification")`. Ad esempio, la query seguente restituisce un'eccezione:

```
g.with("Neptune#ml.endpoint","edge-classification-social-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by", "other_label").with("Neptune#ml.classification")
```

Per i messaggi di errore specifici, consulta Elenco delle eccezioni per le query di inferenza Gremlin in Neptune ML.

È possibile usare un passaggio `properties().with("Neptune#ml.classification")` in combinazione con uno dei seguenti passaggi:

- `value()`
- `value().is()`
- `hasValue()`
- `has(value,"")`
- `key()`
- `key().is()`
- `hasKey()`
- `has(key,"")`
- `path()`

Utilizzo dell'inferenza induttiva in una query di classificazione degli archi

Si supponga di dover aggiungere un nuovo arco a un grafo esistente in un notebook Jupyter, in questo modo:

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

È quindi possibile utilizzare una query di inferenza induttiva per ottenere una valutazione che tenga conto del nuovo arco:

```
%%gremlin
g.with("Neptune#ml.endpoint", "ec-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("scale", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
```

Poiché la query è non deterministica, potrebbe restituire risultati leggermente diversi se la si esegue più volte, in base al neighborhood casuale:

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.21365921]
```

Se sono necessari risultati più coerenti, puoi rendere la query deterministica:

```
%%gremlin
g.with("Neptune#ml.endpoint", "ec-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("scale", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Ora i risultati saranno più o meno gli stessi ogni volta che si esegue la query:

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]
```

Query di regressione degli archi Gremlin in Neptune ML

La regressione degli archi è simile alla classificazione degli archi, tranne per il fatto che il valore dedotto dal modello di machine learning è numerico. Per la regressione degli archi, Neptune ML supporta le stesse query utilizzate per la classificazione.

Gli aspetti principali da notare sono:

- È necessario usare il predicato ML `"Neptune#ml.regression"` per configurare il passaggio `properties()` per questo caso d'uso.
- I predicati `"Neptune#ml.limit"` e `"Neptune#ml.threshold"` non sono applicabili in questo caso d'uso.

- Per filtrare in base al valore, è necessario specificare il valore come numerico.

Sintassi di una query di regressione degli archi Gremlin

Per un grafo semplice in cui `User` è il nodo di testa, `Movie` è il nodo di coda e `Rated` è l'arco che li collega, ecco un esempio di query di regressione degli archi che trova il valore di valutazione numerico, qui denominato punteggio, per l'arco `Rated`:

```
g.with("Neptune#ml.endpoint","edge-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("rating_1","rating_2","rating_3")
 .properties("score").with("Neptune#ml.regression")
```

Puoi anche filtrare in base a un valore dedotto dal modello di regressione ML. Per gli archi `Rated` esistenti (da `User` a `Movie`) identificati da `"rating_1"`, `"rating_2"` e `"rating_3"`, dove la proprietà dell'arco `Score` sia presente per queste classificazioni, è possibile usare una query come la seguente per dedurre `Score` per gli archi in cui è maggiore o uguale a 9:

```
g.with("Neptune#ml.endpoint","edge-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("rating_1","rating_2","rating_3")
 .properties("score").with("Neptune#ml.regression")
 .value().is(P.gte(9))
```

Utilizzo dell'inferenza induttiva in una query di regressione degli archi

Si supponga di dover aggiungere un nuovo arco a un grafo esistente in un notebook Jupyter, in questo modo:

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

È quindi possibile utilizzare una query di inferenza induttiva per ottenere un punteggio che tenga conto del nuovo arco:

```
%%gremlin
g.with("Neptune#ml.endpoint", "er-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
```

```
 .E('e101').properties("score")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
```

Poiché la query è non deterministica, potrebbe restituire risultati leggermente diversi se la si esegue più volte, in base al neighborhood casuale:

```
# First time
==>ep[score->96]

# Second time
==>ep[score->91]
```

Se sono necessari risultati più coerenti, puoi rendere la query deterministica:

```
%%gremlin
g.with("Neptune#ml.endpoint", "er-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("score")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Ora i risultati saranno più o meno gli stessi ogni volta che si esegue la query:

```
# First time
==>ep[score->96]

# Second time
==>ep[score->96]
```

Query di previsione dei collegamenti Gremlin che utilizzano modelli di previsione dei collegamenti in Neptune ML

I modelli di previsione dei collegamenti possono risolvere problemi come i seguenti:

- Previsione del nodo di testa: dati un vertice e un tipo di arco, quali sono i vertici da cui provengono i collegamenti di tale vertice?
- Previsione del nodo di coda: dati un vertice e un tipo di arco, quali sono i vertici a cui si collega tale vertice?

> **Note**
>
> La previsione degli archi non è ancora supportata in Neptune ML.

Per gli esempi seguenti, si consideri un semplice grafo con i vertici `User` e `Movie` collegati dall'arco `Rated`.

Ecco un esempio di query di previsione del nodo di testa, usata per prevedere i primi cinque utenti con le maggiori probabilità di valutare i film, `"movie_1"`, `"movie_2"` e `"movie_3"`:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .with("Neptune#ml.limit", 5)
 .V("movie_1", "movie_2", "movie_3")
 .in("rated").with("Neptune#ml.prediction").hasLabel("user")
```

Eccone una simile per la previsione del nodo di coda, usata per prevedere i primi cinque film con le maggiori probabilità di essere valutati dall'utente `"user_1"`:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out("rated").with("Neptune#ml.prediction").hasLabel("movie")
```

L'etichetta dell'arco e l'etichetta del vertice previsto sono obbligatorie. Se una delle due viene omessa, verrà generata un'eccezione. Ad esempio, la seguente query senza un'etichetta del vertice previsto genera un'eccezione:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out("rated").with("Neptune#ml.prediction")
```

Analogamente, la seguente query senza un'etichetta dell'arco genera un'eccezione:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out().with("Neptune#ml.prediction").hasLabel("movie")
```

Per i messaggi di errore specifici restituiti da queste eccezioni, consulta l'elenco delle eccezioni di Neptune ML.

Altre query di previsione dei collegamenti

Puoi usare il passaggio `select()` con il passaggio `as()` per generare i vertici previsti insieme ai vertici di input:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1").as("source")
 .in("rated").with("Neptune#ml.prediction").hasLabel("user").as("target")
 .select("source","target")

g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1").as("source")
 .out("rated").with("Neptune#ml.prediction").hasLabel("movie").as("target")
 .select("source","target")
```

Puoi eseguire query illimitate, come queste:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out("rated").with("Neptune#ml.prediction").hasLabel("movie")

g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1")
 .in("rated").with("Neptune#ml.prediction").hasLabel("user")
```

Utilizzo dell'inferenza induttiva in una query di previsione dei collegamenti

Si supponga di dover aggiungere un nuovo nodo a un grafo esistente in un notebook Jupyter, in questo modo:

```
%%gremlin
g.addV('label1').property(id,'101').as('newV1')
 .addV('label2').property(id,'102').as('newV2')
 .V('1').as('oldV1')
 .V('2').as('oldV2')
```

```
  .addE('eLabel1').from('newV1').to('oldV1')
  .addE('eLabel2').from('oldV2').to('newV2')
```

È quindi possibile utilizzare una query di inferenza induttiva per prevedere il nodo di testa, tenendo conto del nuovo nodo:

```
%%gremlin
g.with("Neptune#ml.endpoint", "lp-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').out("eLabel1")
 .with("Neptune#ml.prediction")
 .with("Neptune#ml.inductiveInference")
 .hasLabel("label2")
```

Risultato:

```
==>V[2]
```

Allo stesso modo, è possibile utilizzare una query di inferenza induttiva per prevedere il nodo di coda, tenendo conto del nuovo nodo:

```
%%gremlin
g.with("Neptune#ml.endpoint", "lp-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('102').in("eLabel2")
 .with("Neptune#ml.prediction")
 .with("Neptune#ml.inductiveInference")
 .hasLabel("label1")
```

Risultato:

```
==>V[1]
```

Elenco delle eccezioni per le query di inferenza Gremlin in Neptune ML

- **`BadRequestException`**: non è possibile caricare le credenziali per il ruolo fornito.

  Messaggio: `Unable to load credentials for role: the specified IAM Role ARN.`
- **`BadRequestException`**: il ruolo IAM specificato non è autorizzato a richiamare l'endpoint SageMaker.

Messaggio: `User: the specified IAM Role ARN is not authorized to perform: sagemaker:InvokeEndpoint on resource: the specified endpoint.`

- **`BadRequestException`**: l'endpoint specificato non esiste.

  Messaggio: `Endpoint the specified endpoint not found.`

- **`InternalFailureException`**: non è possibile recuperare i metadati di inferenza induttiva in tempo reale di Neptune ML da Amazon S3.

  Messaggio: `Unable to fetch Neptune ML - Real-Time Inductive Inference metadata from S3. Check the permissions of the S3 bucket or if the Neptune instance can connect to S3.`

- **`InternalFailureException`**: Neptune ML non trova il file dei metadati per l'inferenza induttiva in tempo reale in Amazon S3.

  Messaggio: `Neptune ML cannot find the metadata file for Real-Time Inductive Inference in S3.`

- **`InvalidParameterException`**: l'endpoint specificato non è valido dal punto di vista sintattico.

  Messaggio: `Invalid endpoint provided for external service query.`

- **`InvalidParameterException`**: il nome ARN del ruolo IAM per l'esecuzione SageMaker specificata non è valido dal punto di vista sintattico.

  Messaggio: `Invalid IAM role ARN provided for external service query.`

- **`InvalidParameterException`**: nel passaggio `properties()` di una query sono specificate più chiavi delle proprietà.

  Messaggio: `ML inference queries are currently supported for one property key.`

- **`InvalidParameterException`**: in una query vengono specificate più etichette degli archi.

  Messaggio: `ML inference are currently supported only with one edge label.`

- **`InvalidParameterException`**: in una query sono specificati più vincoli per le etichette dei vertici.

  Messaggio: `ML inference are currently supported only with one vertex label constraint.`

- **InvalidParameterException**: nella stessa query sono presenti entrambi i predicati `Neptune#ml.classification` e `Neptune#ml.regression`.

  Messaggio: `Both regression and classification ML predicates cannot be specified in the query.`
- **InvalidParameterException**: è stata specificata più di un'etichetta dell'arco nel passaggio `in()` o `out()` di una query di previsione dei collegamenti.

  Messaggio: `ML inference are currently supported only with one edge label.`
- **InvalidParameterException**: è stata specificata più di una chiave delle proprietà con Neptune#ml.score.

  Messaggio: `Neptune ML inference queries are currently supported for one property key and one Neptune#ml.score property key.`
- **MissingParameterException**: l'endpoint non è stato specificato nella query o come parametro del cluster database.

  Messaggio: `No endpoint provided for external service query.`
- MissingParameterException: il ruolo IAM dell'esecuzione SageMaker non è stato specificato nella query o come parametro del cluster database.

  Messaggio: `No IAM role ARN provided for external service query.`
- **MissingParameterException**: la chiave della proprietà non è presente nel passaggio `properties()` in una query.

  Messaggio: `Property key needs to be specified using properties() step for ML inference queries.`
- **MissingParameterException**: non è stata specificata alcuna un'etichetta dell'arco nel passaggio `in()` o `out()` di una query di previsione dei collegamenti.

  Messaggio: `Edge label needs to be specified while using in() or out() step for ML inference queries.`
- **MissingParameterException**: non è stata specificata alcuna chiave della proprietà con Neptune#ml.score.

  Messaggio: `Property key needs to be specified along with Neptune#ml.score property key while using the properties() step for Neptune ML inference queries.`

- **UnsupportedOperationException**: in una query di previsione dei collegamenti è stato usato un passaggio `both()`.

  Messaggio: `ML inference queries are currently not supported with both() step.`
- **UnsupportedOperationException**: non è stata specificata alcuna etichetta del vertice previsto nel passaggio `has()` con il passaggio `in()` o `out()` in una query di previsione dei collegamenti.

  Messaggio: `Predicted vertex label needs to be specified using has() step for ML inference queries.`
- **UnsupportedOperationException**: le query di inferenza induttiva ML Gremlin non sono attualmente supportate con passaggi non ottimizzati.

  Messaggio: `Neptune ML - Real-Time Inductive Inference queries are currently not supported with Gremlin steps which are not optimized for Neptune. Check the Neptune User Guide for a list of Neptune-optimized steps.`
- **UnsupportedOperationException**: le query di inferenza di Neptune ML non sono attualmente supportate all'interno di un passaggio `repeat`.

  Messaggio: `Neptune ML inference queries are currently not supported inside a repeat step.`
- **UnsupportedOperationException**: attualmente non sono supportate più query di inferenza Neptune ML per ogni query Gremlin.

  Messaggio: `Neptune ML inference queries are currently supported only with one ML inference query per gremlin query.`

Query di inferenza SPARQL in Neptune ML

Neptune ML mappa il grafo RDF in un grafo delle proprietà per modellare l'attività di machine learning. Attualmente supporta i seguenti casi d'uso:

- Classificazione degli oggetti: prevede la funzionalità categoriale di un oggetto.
- Regressione degli oggetti: prevede una proprietà numerica di un oggetto.
- Previsione dell'oggetto: prevede un oggetto in base a un soggetto e a una relazione.
- Previsione del soggetto: prevede un oggetto in base a un soggetto e a una relazione.

> **Note**
>
> Neptune ML non supporta i casi d'uso di classificazione e regressione dei soggetti con SPARQL.

Predicati Neptune ML utilizzati nelle query di inferenza SPARQL

Con l'inferenza SPARQL vengono usati i predicati seguenti:

Predicato **`neptune-ml:timeout`**

Specifica il timeout per la connessione al server remoto. Non deve essere confuso con il timeout della richiesta di query, che è il tempo massimo che il server può impiegare per soddisfare una richiesta.

Tieni presente che se il timeout della query si verifica prima del timeout del servizio specificato dal predicato `neptune-ml:timeout`, viene annullata anche la connessione al servizio.

Predicato **`neptune-ml:outputClass`**

Il predicato `neptune-ml:outputClass` viene utilizzato solo per definire la classe dell'oggetto previsto per la previsione dell'oggetto o del soggetto previsto per la previsione del soggetto.

Predicato **`neptune-ml:outputScore`**

Il predicato `neptune-ml:outputScore` è un numero positivo che rappresenta la probabilità che l'output di un modello di machine learning sia corretto.

Predicato **`neptune-ml:modelType`**

Il predicato `neptune-ml:modelType` specifica il tipo di modello di machine learning da addestrare:

- `OBJECT_CLASSIFICATION`
- `OBJECT_REGRESSION`
- `OBJECT_PREDICTION`
- `SUBJECT_PREDICTION`

Predicato **`neptune-ml:input`**

Il predicato `neptune-ml:input` fa riferimento all'elenco degli URI utilizzati come input per Neptune ML.

Predicato **`neptune-ml:output`**

Il predicato `neptune-ml:output` fa riferimento all'elenco dei set di associazione in cui Neptune ML restituisce risultati.

Predicato **`neptune-ml:predicate`**

Il predicato `neptune-ml:predicate` predicato viene usato in modo diverso a seconda dell'attività da eseguire:

- Per la previsione di oggetti o soggetti: definisce il tipo di predicato (tipo di arco o relazione).
- Per la classificazione e la regressione degli oggetti: definisce il valore letterale (proprietà) da prevedere.

Predicato **`neptune-ml:batchSize`**

`neptune-ml:batchSize` specifica le dimensioni di input per la chiamata al servizio remoto.

Esempi di classificazione degli oggetti SPARQL

Per la classificazione degli oggetti SPARQL in Neptune ML, il modello viene addestrato su uno dei valori del predicato. È utile quando tale predicato non è già presente in un determinato soggetto.

Con il modello di classificazione degli oggetti è possibile dedurre solo i valori dei predicati categoriali.

La seguente query tenta di prevedere il valore del predicato <http://www.example.org/team> per tutti gli input di tipo `foaf:Person`:

```
SELECT * WHERE { ?input a foaf:Person .
```

```
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_CLASSIFICATION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/team> ;
                      neptune-ml:output ?output .
  }
}
```

Questa query può essere personalizzata nel modo seguente:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-account-balance-endpoint' ;
                      neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                      neptune-ml:batchSize "40"^^xsd:integer ;
                      neptune-ml:timeout "1000"^^xsd:integer ;

                      neptune-ml:modelType 'OBJECT_CLASSIFICATION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/team> ;
                      neptune-ml:output ?output .
  }
}
```

Esempi di regressione degli oggetti SPARQL

La regressione degli oggetti è simile alla classificazione degli oggetti, ad eccezione del valore numerico del predicato dedotto dal modello di regressione per ogni nodo. È possibile usare le stesse query SPARQL usate per la classificazione degli oggetti anche per la regressione degli oggetti, con l'eccezione che i predicati `the Neptune#ml.limit` e `Neptune#ml.threshold` non sono applicabili.

La seguente query tenta di prevedere il valore del predicato <http://www.example.org/accountbalance> per tutti gli input di tipo `foaf:Person`:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_REGRESSION' ;
                      neptune-ml:input ?input ;
```

```
                    neptune-ml:predicate <http://www.example.org/accountbalance> ;
                    neptune-ml:output ?output .
  }
}
```

Questa query può essere personalizzata nel modo seguente:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-account-balance-endpoint' ;
                    neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                    neptune-ml:batchSize "40"^^xsd:integer ;
                    neptune-ml:timeout "1000"^^xsd:integer ;

                    neptune-ml:modelType 'OBJECT_REGRESSION' ;
                    neptune-ml:input ?input ;
                    neptune-ml:predicate <http://www.example.org/accountbalance> ;
                    neptune-ml:output ?output .
  }
}
```

Esempio di previsione degli oggetti SPARQL

La previsione degli oggetti prevede il valore dell'oggetto per un determinato soggetto e predicato.

La seguente query di previsione degli oggetti tenta di prevedere il tipo di film che piacerà all'input di tipo `foaf:Person`:

```
?x a foaf:Person .
?x   <http://www.example.org/likes> ?m .
?m a <http://www.example.org/movie> .

## Query
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_PREDICTION' ;
                    neptune-ml:input ?input ;
                    neptune-ml:predicate <http://www.example.org/likes> ;
                    neptune-ml:output ?output ;
                    neptune-ml:outputClass <http://www.example.org/movie> .
```

```
  }
}
```

La query stessa può essere personalizzata come segue:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-user-movie-prediction-
endpoint' ;
                      neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                      neptune-ml:limit "5"^^xsd:integer ;
                      neptune-ml:batchSize "40"^^xsd:integer ;
                      neptune-ml:threshold "0.1"^^xsd:double ;
                      neptune-ml:timeout "1000"^^xsd:integer ;
                      neptune-ml:outputScore ?score ;

                      neptune-ml:modelType 'OBJECT_PREDICTION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/likes> ;
                      neptune-ml:output ?output ;
                      neptune-ml:outputClass <http://www.example.org/movie> .
  }
}
```

Esempio di previsione dei soggetti SPARQL

La previsione dei soggetti prevede il valore del soggetto per un determinato predicato e oggetto.

Ad esempio, la seguente query prevede chi (di tipo `foaf:User`) guarderà un determinato film:

```
SELECT * WHERE { ?input (a foaf:Movie) .
    SERVICE neptune-ml:inference {
        neptune-ml:config neptune-ml:modelType 'SUBJECT_PREDICTION' ;
                          neptune-ml:input ?input ;
                          neptune-ml:predicate <http://aws.amazon.com/neptune/csv2rdf/
object_Property/rated> ;
                          neptune-ml:output ?output ;
                          neptune-ml:outputClass <http://aws.amazon.com/neptune/
csv2rdf/class/User> ;        }
}
```

Elenco delle eccezioni per le query di inferenza SPARQL in Neptune ML

- **BadRequestException** – Messaggio: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects at least 1 value for the parameter *(parameter name)*, found zero.
- **BadRequestException** – Messaggio: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects at most 1 value for the parameter *(parameter name)*, found *(a number)* values.
- **BadRequestException** – Messaggio: Invalid predicate *(predicate name)* provided for external service http://aws.amazon.com/neptune/vocab/v01/services/ml#inference query.
- **BadRequestException** – Messaggio: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the predicate *(predicate name)* to be defined.
- **BadRequestException** – Messaggio: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the value of (parameter) *(parameter name)* to be a variable, found: *(type)*"
- **BadRequestException** – Messaggio: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the input *(parameter name)* to be a constant, found: *(type)*.
- **BadRequestException** – Messaggio: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference is expected to return only 1 value.
- **BadRequestException** – Messaggio: "The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference only allows StatementPatternNodes.
- **BadRequestException** – Messaggio: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference does not allow the predicate *(predicate name)*.
- **BadRequestException** – Messaggio: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference predicates cannot be variables, found: *(type)*.

- **`BadRequestException`** – Messaggio: `The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference predicates are expected to be part of the namespace (namespace name), found: (namespace name).`

Documentazione di riferimento per l'API di gestione Neptune ML

Indice

- Errori ed eccezioni dell'API di gestione Neptune ML

Elaborazione dei dati tramite il comando **`dataprocessing`**

Si utilizza il comando `dataprocessing` in Neptune ML per creare un processo di elaborazione dei dati, controllarne lo stato, arrestarlo o elencare tutti i processi di elaborazione dei dati attivi.

Creazione di un processo di elaborazione dei dati con il comando **`dataprocessing`** in Neptune ML

Un tipico comando `dataprocessing` in Neptune ML per la creazione di un nuovo processo ha il seguente aspetto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)"
      }'
```

Un comando per avviare una rielaborazione incrementale ha il seguente aspetto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for this job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)"
        "previousDataProcessingJobId" : "(the job ID of a previously completed job to
 update)"
}'
```

Parametri per il comando **`dataprocessing`** per la creazione del processo

- **`id`** (facoltativo): identificatore univoco per il nuovo processo.

  Tipo: stringa. Impostazione predefinita: un UUID generato automaticamente.

- **`previousDataProcessingJobId`** (facoltativo): ID del processo di elaborazione dei dati completato che è stato eseguito su una versione precedente dei dati.

  Tipo: stringa. Impostazione predefinita: none.

  Nota: usa questo parametro per l'elaborazione incrementale dei dati, per aggiornare il modello in caso di modifica dei dati del grafo (ma non quando i dati sono stati eliminati).
- **`inputDataS3Location`** (obbligatorio): URI della posizione Amazon S3 in cui SageMaker deve scaricare i dati necessari per eseguire il processo di elaborazione dei dati.

  Tipo: stringa.
- **`processedDataS3Location`** (obbligatorio): URI della posizione Amazon S3 in cui SageMaker deve scaricare i risultati di un processo di elaborazione dei dati.

  Tipo: stringa.
- **`sagemakerIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM per l'esecuzione di SageMaker.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.
- **`neptuneIamRoleArn`** (facoltativo): nome della risorsa Amazon (ARN) di un ruolo IAM che SageMaker può assumere per eseguire attività per tuo conto.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.
- **`processingInstanceType`** (facoltativo): tipo di istanza ML utilizzata durante l'elaborazione dei dati. La sua memoria deve essere sufficientemente grande da contenere il set di dati elaborato.

  Tipo: stringa. Impostazione predefinita: tipo `ml.r5` più piccolo, la cui memoria è dieci volte più grande delle dimensioni dei dati dei grafi esportati su disco.

  Nota: Neptune ML può selezionare automaticamente il tipo di istanza. Per informazioni, consultare Selezione di un'istanza per l'elaborazione dei dati.
- **`processingInstanceVolumeSizeInGB`** (facoltativo): dimensioni del volume del disco dell'istanza di elaborazione. Sia i dati di input che i dati elaborati vengono archiviati su disco, quindi le dimensioni del volume devono essere sufficientemente grandi da contenere entrambi i set di dati.

  Tipo: numero intero. Default: `0`

Nota: se non è specificato o se è pari a 0, Neptune ML sceglie automaticamente le dimensioni del volume in base a quelle dei dati.

- **`processingTimeOutInSeconds`** (facoltativo): timeout in secondi per il processo di elaborazione dei dati.

  Tipo: numero intero. Impostazione predefinita: `86,400` (7 giorni)
- **`modelType`** (facoltativo): uno dei due tipi di modello attualmente supportati da Neptune ML: modelli a grafo eterogenei (`heterogeneous`) e grafo della conoscenza (`kge`).

  Tipo: stringa. Impostazione predefinita: none.

  Nota: se non è specificato, Neptune ML sceglie automaticamente il tipo di modello in base ai dati.
- **`configFileName`** (facoltativo): file di specifica dei dati che descrive come caricare i dati dei grafi esportati per l'addestramento. Il file viene generato automaticamente dal kit di strumenti di esportazione Neptune.

  Tipo: stringa. Default: `training-data-configuration.json`
- **`subnets`** (facoltativo): ID delle sottoreti nel VPC Neptune.

  Tipo: elenco di stringhe. Impostazione predefinita: none.
- **`securityGroupIds`** (facoltativo): ID del gruppo di sicurezza del VPC.

  Tipo: elenco di stringhe. Impostazione predefinita: none.
- **`volumeEncryptionKMSKey`** (facoltativo): chiave AWS Key Management Service (AWS KMS) usata da SageMaker per crittografare i dati nel volume di archiviazione collegato alle istanze di calcolo ML che eseguono il processo di elaborazione.

  Tipo: stringa. Impostazione predefinita: none.
- **`enableInterContainerTrafficEncryption`** (facoltativo): abilita o disabilita la crittografia del traffico tra container nei processi di addestramento o di ottimizzazione degli iperparametri.

  Tipo: booleano. Impostazione predefinita: True.

> **Note**
>
> Il parametro `enableInterContainerTrafficEncryption` è disponibile solo nel rilascio del motore 1.2.0.2.R3.

- **`s3OutputEncryptionKMSKey`** (facoltativo): chiave AWS Key Management Service (AWS KMS) utilizzata da SageMaker per crittografare l'output del processo di addestramento.

  Tipo: stringa. Impostazione predefinita: none.

Recupero dello stato di un processo di elaborazione dei dati con il comando **`dataprocessing`** in Neptune ML

Un comando `dataprocessing` di esempio in Neptune ML per lo stato di un processo ha il seguente aspetto:

```
curl -s \
  "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)" \
  | python -m json.tool
```

Parametri per il comando **`dataprocessing`** per lo stato del processo

- **`id`** (obbligatorio): identificatore univoco del processo di elaborazione dei dati.

  Tipo: stringa.
- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Arresto di un processo di elaborazione dei dati con il comando **`dataprocessing`** in Neptune ML

Un comando `dataprocessing` di esempio in Neptune ML per l'arresto di un processo ha il seguente aspetto:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)"
```

Oppure questo:

```
curl -s \
```

```
  -X DELETE "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)?clean=true"
```

Parametri per il comando **`dataprocessing`** per l'arresto del processo

- **`id`** (obbligatorio): identificatore univoco del processo di elaborazione dei dati.

  Tipo: stringa.
- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.
- **`clean`** (facoltativo): questo flag specifica che tutti gli artefatti Amazon S3 devono essere eliminati quando il processo viene arrestato.

  Tipo: Boolean Default: `FALSE`

Elenco dei processi di elaborazione dei dati attivi con il comando **`dataprocessing`** in Neptune ML.

Un comando `dataprocessing` di esempio in Neptune ML per l'elenco dei processi attivi ha il seguente aspetto:

```
curl -s "https://(your Neptune endpoint)/ml/dataprocessing"
```

Oppure questo:

```
curl -s "https://(your Neptune endpoint)/ml/dataprocessing?maxItems=3"
```

Parametri per il comando **`dataprocessing`** per l'elenco dei processi

- **`maxItems`** (facoltativo): numero massimo di elementi da restituire.

  Tipo: numero intero. Default: `10` Valore massimo consentito: `1024`.
- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Addestramento dei modelli con il comando **`modeltraining`**

È possibile utilizzare il comando `modeltraining` in Neptune ML per creare un processo di addestramento dei modelli, controllarne lo stato, arrestarlo o elencare tutti i processi di addestramento dei modelli attivi.

Creazione di un processo di addestramento dei modelli con il comando **`modeltraining`** in Neptune ML

Un comando `modeltraining` in Neptune ML per la creazione di un processo completamente nuovo ha il seguente aspetto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
      }'
```

Un comando `modeltraining` in Neptune ML per la creazione di un processo di aggiornamento per l'addestramento dei modelli incrementale ha il seguente aspetto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "previousModelTrainingJobId" : "(the job ID of a completed model-training job
 to update)",
      }'
```

Un comando `modeltraining` in Neptune ML per la creazione di un nuovo processo con l'implementazione di un modello personalizzato fornita dall'utente ha il seguente aspetto:

```
curl \
```

```
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Parametri per il comando **`modeltraining`** per la creazione del processo

- **`id`** (facoltativo): identificatore univoco per il nuovo processo.

  Tipo: stringa. Impostazione predefinita: un UUID generato automaticamente.
- **`dataProcessingJobId`** (obbligatorio): ID del processo di elaborazione dei dati completato che ha creato i dati per l'addestramento.

  Tipo: stringa.
- **`trainModelS3Location`** (obbligatorio): posizione in Amazon S3 in cui devono essere archiviati gli artefatti del modello.

  Tipo: stringa.
- **`previousModelTrainingJobId`** (facoltativo): ID del processo di addestramento del modello completato che si desidera aggiornare in modo incrementale in base ai dati aggiornati.

  Tipo: stringa. Impostazione predefinita: none.
- **`sagemakerIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM per l'esecuzione di SageMaker.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

- **`modelName`** (facoltativo): tipo di modello per l'addestramento. Per impostazione predefinita, il modello di machine learning è basato automaticamente sul `modelType` utilizzato nell'elaborazione dei dati, ma qui è possibile specificare un tipo di modello diverso.

  Tipo: stringa. Impostazione predefinita: `rgcn` per i grafi eterogenei e `kge` per i grafi della conoscenza. Valori validi: per i grafi eterogenei: `rgcn`. Per i grafi `kge`: `transe`, `distmult` o `rotate`. Per l'implementazione di un modello personalizzato: `custom`.

- **`baseProcessingInstanceType`** (facoltativo): tipo di istanza ML utilizzata nella preparazione e nella gestione dell'addestramento dei modelli ML.

  Tipo: stringa. Nota: si tratta di un'istanza CPU scelta in base ai requisiti di memoria per l'elaborazione dei dati e del modello di addestramento. Per informazioni, consultare Selezione di un'istanza per l'addestramento e la trasformazione del modello.

- **`trainingInstanceType`** (facoltativo): tipo di istanza ML utilizzato per l'addestramento del modello. Tutti i modelli Neptune ML supportano l'addestramento su CPU, GPU e multiGPU.

  Tipo: stringa. Default: `ml.p3.2xlarge`

  Nota: la scelta del tipo di istanza corretto per l'addestramento dipende dal tipo di attività, dalle dimensioni del grafo e dal budget. Per informazioni, consultare Selezione di un'istanza per l'addestramento e la trasformazione del modello.

- **`trainingInstanceVolumeSizeInGB`** (facoltativo): dimensioni del volume del disco dell'istanza di addestramento. Sia i dati di input che il modello di output vengono archiviati su disco, quindi le dimensioni del volume devono essere sufficientemente grandi da contenere entrambi i set di dati.

  Tipo: numero intero. Default: `0`

  Nota: se non è specificato o è pari a 0, Neptune ML seleziona le dimensioni del volume del disco in base alla raccomandazione generata nella fase di elaborazione dati. Per informazioni, consultare Selezione di un'istanza per l'addestramento e la trasformazione del modello.

- **`trainingTimeOutInSeconds`** (facoltativo): timeout in secondi per il processo di addestramento.

  Tipo: numero intero. Impostazione predefinita: `86,400` (7 giorni)

- **`maxHPONumberOfTrainingJobs`**: numero totale massimo di processi di addestramento da avviare per il processo di ottimizzazione degli iperparametri.

  Tipo: numero intero. Default: 2

  Nota: Neptune ML ottimizza automaticamente gli iperparametri del modello di machine learning. Per ottenere un modello che funzioni correttamente, utilizza almeno 10 processi (in altre parole, imposta `maxHPONumberOfTrainingJobs` su 10). In generale, maggiore è il numero di esecuzioni di ottimizzazione, migliori sono i risultati.

- **`maxHPOParallelTrainingJobs`**: Numero massimo di processi di addestramento paralleli da avviare per il processo di ottimizzazione degli iperparametri.

  Tipo: numero intero. Default: 2

  Nota: il numero di processi paralleli che è possibile eseguire è limitato dalle risorse disponibili sull'istanza di addestramento.

- **`subnets`** (facoltativo): ID delle sottoreti nel VPC Neptune.

  Tipo: elenco di stringhe. Impostazione predefinita: none.

- **`securityGroupIds`** (facoltativo): ID del gruppo di sicurezza del VPC.

  Tipo: elenco di stringhe. Impostazione predefinita: none.

- **`volumeEncryptionKMSKey`** (facoltativo): chiave AWS Key Management Service (AWS KMS) usata da SageMaker per crittografare i dati nel volume di archiviazione collegato alle istanze di calcolo ML che eseguono il processo di addestramento.

  Tipo: stringa. Impostazione predefinita: none.

- **`s3OutputEncryptionKMSKey`** (facoltativo): chiave AWS Key Management Service (AWS KMS) utilizzata da SageMaker per crittografare l'output del processo di elaborazione.

  Tipo: stringa. Impostazione predefinita: none.

- **`enableInterContainerTrafficEncryption`** (facoltativo): abilita o disabilita la crittografia del traffico tra container nei processi di addestramento o di ottimizzazione degli iperparametri.

  Tipo: booleano. Impostazione predefinita: True.

> **Note**
>
> Il parametro `enableInterContainerTrafficEncryption` è disponibile solo nel rilascio del motore 1.2.0.2.R3.

- **`enableManagedSpotTraining`** (facoltativo): ottimizza il costo per l'addestramento dei modelli di machine learning utilizzando le istanze spot di Amazon Elastic Compute Cloud. Per ulteriori informazioni, consulta Managed Spot Training in Amazon SageMaker.

  Tipo: Boolean Impostazione predefinita: False.

- **`customModelTrainingParameters`** (facoltativo): configurazione per l'addestramento di modelli personalizzati. Si tratta di un oggetto JSON con i campi seguenti:

  - **`sourceS3DirectoryPath`** (obbligatorio): percorso della posizione Amazon S3 in cui si trova il modulo Python che implementa il modello. Deve indicare una posizione Amazon S3 esistente valida che contenga almeno uno script di addestramento, uno script di trasformazione e un file `model-hpo-configuration.json`.
  - **`trainingEntryPointScript`** (facoltativo): nome del punto di ingresso nel modulo di uno script che esegue l'addestramento del modello e accetta gli iperparametri come argomenti della riga di comando, inclusi gli iperparametri fissi.

    Default: `training.py`
  - **`transformEntryPointScript`** (facoltativo): nome del punto di ingresso nel modulo di uno script che deve essere eseguito dopo aver identificato il modello migliore ottenuto dalla ricerca degli iperparametri, in modo da calcolare gli artefatti del modello necessari per l'implementazione del modello. Deve poter essere eseguito senza argomenti della riga di comando.

    Default: `transform.py`

- **`maxWaitTime`** (facoltativo): tempo massimo di attesa, in secondi, quando si esegue l'addestramento dei modelli utilizzando istanze spot. Deve essere maggiore di `trainingTimeOutInSeconds`.

  Tipo: numero intero.

Recupero dello stato di un processo di addestramento dei modelli con il comando **`modeltraining`** in Neptune ML

Un comando `modeltraining` di esempio in Neptune ML per lo stato di un processo ha il seguente aspetto:

```
curl -s \
  "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)" \
  | python -m json.tool
```

Parametri per il comando **`modeltraining`** per lo stato del processo

- **`id`** (obbligatorio): identificatore univoco del processo di addestramento dei modelli.

  Tipo: stringa.
- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Arresto di un processo di addestramento dei modelli con il comando **`modeltraining`** in Neptune ML

Un comando `modeltraining` di esempio in Neptune ML per l'arresto di un processo ha il seguente aspetto:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)"
```

Oppure questo:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)?clean=true"
```

Parametri per il comando **`modeltraining`** per l'arresto del processo

- **`id`** (obbligatorio): identificatore univoco del processo di addestramento dei modelli.

Tipo: stringa.

- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.
- **`clean`** (facoltativo): questo flag specifica che tutti gli artefatti Amazon S3 devono essere eliminati quando il processo viene arrestato.

  Tipo: Boolean Default: `FALSE`

Elenco dei processi di addestramento dei modelli attivi con il comando **`modeltraining`** in Neptune ML

Un comando `modeltraining` di esempio in Neptune ML per l'elenco dei processi attivi ha il seguente aspetto:

```
curl -s "https://(your Neptune endpoint)/ml/modeltraining" | python -m json.tool
```

Oppure questo:

```
curl -s "https://(your Neptune endpoint)/ml/modeltraining?maxItems=3" | python -m
 json.tool
```

Parametri per il comando **`modeltraining`** per l'elenco dei processi

- **`maxItems`** (facoltativo): numero massimo di elementi da restituire.

  Tipo: numero intero. Default: `10` Valore massimo consentito: `1024`.
- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Trasformazione dei modelli con il comando **`modeltransform`**

È possibile utilizzare il comando `modeltransform` in Neptune ML per creare un processo di trasformazione dei modelli, controllarne lo stato, arrestarlo o elencare tutti i processi di trasformazione dei modelli attivi.

Creazione di un processo di trasformazione dei modelli con il comando **`modeltransform`** in Neptune ML

Un comando `modeltransform` in Neptune ML per la creazione di un processo di trasformazione incrementale senza riaddestramento dei modelli ha il seguente aspetto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "dataProcessingJobId" : "(the job-id of a completed data-processing job)",
        "mlModelTrainingJobId" : "(the job-id of a completed model-training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform"
      }'
```

Un comando `modeltransform` in Neptune ML per la creazione di un processo da un processo di addestramento SageMaker completato ha il seguente aspetto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform",
        "baseProcessingInstanceType" : ""
      }'
```

Un comando `modeltransform` in Neptune ML per la creazione di un processo che usa l'implementazione di un modello personalizzato ha il seguente aspetto:

```
curl \
```

```
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
        "customModelTransformParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Parametri per il comando **`modeltransform`** per la creazione del processo

- **`id`** (facoltativo): identificatore univoco per il nuovo processo.

  Tipo: stringa. Impostazione predefinita: un UUID generato automaticamente.
- **`dataProcessingJobId`**: ID di un processo di elaborazione dei dati completato.

  Tipo: stringa.

  Nota: è necessario includere sia `dataProcessingJobId` che `mlModelTrainingJobId`, o `trainingJobName`.
- **`mlModelTrainingJobId`**: ID di un processo di addestramento del modello completato.

  Tipo: stringa.

  Nota: è necessario includere sia `dataProcessingJobId` che `mlModelTrainingJobId`, o `trainingJobName`.
- **`trainingJobName`**: nome di un processo di addestramento SageMaker completato.

  Tipo: stringa.

  Nota: è necessario includere entrambi i parametri `dataProcessingJobId` e `mlModelTrainingJobId` oppure il parametro `trainingJobName`.
- **`sagemakerIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM per l'esecuzione di SageMaker.

Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

- **`customModelTransformParameters`** (facoltativo): informazioni di configurazione per una trasformazione del modello mediante un modello personalizzato. L'oggetto `customModelTransformParameters` contiene i seguenti campi, che devono avere valori compatibili con i parametri del modello salvati durante il processo di addestramento:

  - **`sourceS3DirectoryPath`** (obbligatorio): percorso della posizione Amazon S3 in cui si trova il modulo Python che implementa il modello. Deve indicare una posizione Amazon S3 esistente valida che contenga almeno uno script di addestramento, uno script di trasformazione e un file `model-hpo-configuration.json`.
  - **`transformEntryPointScript`** (facoltativo): nome del punto di ingresso nel modulo di uno script che deve essere eseguito dopo aver identificato il modello migliore ottenuto dalla ricerca degli iperparametri, in modo da calcolare gli artefatti del modello necessari per l'implementazione del modello. Deve poter essere eseguito senza argomenti della riga di comando.

    Default: `transform.py`

- **`baseProcessingInstanceType`** (facoltativo): tipo di istanza ML utilizzata nella preparazione e nella gestione dell'addestramento dei modelli ML.

  Tipo: stringa. Nota: si tratta di un'istanza CPU scelta in base ai requisiti di memoria per l'elaborazione dei dati e del modello di trasformazione. Per informazioni, consultare Selezione di un'istanza per l'addestramento e la trasformazione del modello.

- **`baseProcessingInstanceVolumeSizeInGB`** (facoltativo): dimensioni del volume del disco dell'istanza di addestramento. Sia i dati di input che il modello di output vengono archiviati su disco, quindi le dimensioni del volume devono essere sufficientemente grandi da contenere entrambi i set di dati.

  Tipo: numero intero. Default: `0`

Nota: se non è specificato o è pari a 0, Neptune ML seleziona le dimensioni del volume del disco in base alla raccomandazione generata nella fase di elaborazione dati. Per informazioni, consultare Selezione di un'istanza per l'addestramento e la trasformazione del modello.

- **`subnets`** (facoltativo): ID delle sottoreti nel VPC Neptune.

  Tipo: elenco di stringhe. Impostazione predefinita: none.

- **`securityGroupIds`** (facoltativo): ID del gruppo di sicurezza del VPC.

  Tipo: elenco di stringhe. Impostazione predefinita: none.

- **`volumeEncryptionKMSKey`** (facoltativo): chiave AWS Key Management Service (AWS KMS) usata da SageMaker per crittografare i dati nel volume di archiviazione collegato alle istanze di calcolo ML che eseguono il processo di trasformazione.

  Tipo: stringa. Impostazione predefinita: none.

- **`enableInterContainerTrafficEncryption`** (facoltativo): abilita o disabilita la crittografia del traffico tra container nei processi di addestramento o di ottimizzazione degli iperparametri.

  Tipo: booleano. Impostazione predefinita: True.

  > **Note**
  >
  > Il parametro `enableInterContainerTrafficEncryption` è disponibile solo nel rilascio del motore 1.2.0.2.R3.

- **`s3OutputEncryptionKMSKey`** (facoltativo): chiave AWS Key Management Service (AWS KMS) utilizzata da SageMaker per crittografare l'output del processo di elaborazione.

  Tipo: stringa. Impostazione predefinita: none.

Recupero dello stato di un processo di trasformazione dei modelli con il comando **`modeltransform`** in Neptune ML

Un comando `modeltransform` di esempio in Neptune ML per lo stato di un processo ha il seguente aspetto:

```
curl -s \
  "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)" \
```

```
  | python -m json.tool
```

Parametri per il comando **`modeltransform`** per lo stato del processo

- **`id`** (obbligatorio): identificatore univoco del processo di trasformazione dei modelli.

  Tipo: stringa.
- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Arresto di un processo di trasformazione dei modelli con il comando **`modeltransform`** in Neptune ML

Un comando `modeltransform` di esempio in Neptune ML per l'arresto di un processo ha il seguente aspetto:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)"
```

Oppure questo:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)?clean=true"
```

Parametri per il comando **`modeltransform`** per l'arresto del processo

- **`id`** (obbligatorio): identificatore univoco del processo di trasformazione dei modelli.

  Tipo: stringa.
- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.
- **`clean`** (facoltativo): questo flag specifica che tutti gli artefatti Amazon S3 devono essere eliminati quando il processo viene arrestato.

Tipo: Boolean Default: `FALSE`

Elenco dei processi di trasformazione dei modelli attivi con il comando **`modeltransform`** in Neptune ML

Un comando `modeltransform` di esempio in Neptune ML per l'elenco dei processi attivi ha il seguente aspetto:

```
curl -s "https://(your Neptune endpoint)/ml/modeltransform" | python -m json.tool
```

Oppure questo:

```
curl -s "https://(your Neptune endpoint)/ml/modeltransform?maxItems=3" | python -m
 json.tool
```

Parametri per il comando **`modeltransform`** per l'elenco dei processi

- **`maxItems`** (facoltativo): numero massimo di elementi da restituire.

  Tipo: numero intero. Default: `10` Valore massimo consentito: `1024`.
- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Gestione degli endpoint di inferenza mediante il comando **`endpoints`**

Si utilizza il comando `endpoints` in Neptune ML per creare un endpoint di inferenza, controllarne lo stato, eliminarlo o elencare gli endpoint di inferenza esistenti.

Creazione di un endpoint di inferenza con il comando **`endpoints`** in Neptune ML

Un comando `endpoints` in Neptune ML per la creazione di un endpoint di inferenza da un modello creato da un processo di addestramento ha il seguente aspetto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Un comando `endpoints` in Neptune ML per l'aggiornamento di un endpoint di inferenza esistente da un modello creato da un processo di addestramento ha il seguente aspetto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "update" : "true",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Un comando `endpoints` in Neptune ML per la creazione di un endpoint di inferenza da un modello creato da un processo di trasformazione dei modelli ha il seguente aspetto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTransformJobId": "(the model-training job-id of a completed job)"
      }'
```

Un comando `endpoints` in Neptune ML per l'aggiornamento di un endpoint di inferenza esistente da un modello creato da un processo di trasformazione dei modelli ha il seguente aspetto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "update" : "true",
        "mlModelTransformJobId": "(the model-training job-id of a completed job)"
      }'
```

Parametri per il comando **`endpoints`** per la creazione di endpoint di inferenza

- **`id`** (facoltativo): identificatore univoco per il nuovo endpoint di inferenza.

  Tipo: stringa. Impostazione predefinita: nome con timestamp generato automaticamente.
- **`mlModelTrainingJobId`**: ID del processo di addestramento dei modelli completato che ha creato il modello a cui punterà l'endpoint di inferenza.

  Tipo: stringa.

  Nota: è necessario specificare `mlModelTrainingJobId` o `mlModelTransformJobId`.
- **`mlModelTransformJobId`**: ID del processo di trasformazione dei modelli completato.

  Tipo: stringa.

  Nota: è necessario specificare `mlModelTrainingJobId` o `mlModelTransformJobId`.
- **`update`** (facoltativo): se presente, questo parametro indica che si tratta di una richiesta di aggiornamento.

  Tipo: Boolean Valore predefinito: `false`

  Nota: è necessario specificare `mlModelTrainingJobId` o `mlModelTransformJobId`.
- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

- **`modelName`** (facoltativo): tipo di modello per l'addestramento. Per impostazione predefinita, il modello di machine learning è basato automaticamente sul `modelType` utilizzato nell'elaborazione dei dati, ma qui è possibile specificare un tipo di modello diverso.

  Tipo: stringa. Impostazione predefinita: `rgcn` per i grafi eterogenei e `kge` per i grafi della conoscenza. Valori validi: per i grafi eterogenei: `rgcn`. Per i grafi della conoscenza: `kge`, `transe`, `distmult` o `rotate`.

- **`instanceType`** (facoltativo): tipo di istanza ML utilizzata per l'elaborazione online.

  Tipo: stringa. Default: `ml.m5.xlarge`

  Nota: la scelta dell'istanza ML per un endpoint di inferenza dipende dal tipo di attività, dalle dimensioni del grafo e dal budget. Per informazioni, consultare Selezione di un'istanza per un endpoint di inferenza.

- **`instanceCount`** (facoltativo): numero minimo di istanze Amazon EC2 da implementare su un endpoint ai fini della previsione.

  Tipo: numero intero. Default: 1

- **`volumeEncryptionKMSKey`** (facoltativo): chiave AWS Key Management Service (AWS KMS) usata da SageMaker per crittografare i dati nel volume di archiviazione collegato alle istanze di calcolo ML che eseguono gli endpoint.

  Tipo: stringa. Impostazione predefinita: none.

Recupero dello stato di un endpoint di inferenza con il comando **`endpoints`** in Neptune ML

Un comando `endpoints` di esempio in Neptune ML per lo stato di un endpoint dell'istanza ha il seguente aspetto:

```
curl -s \
  "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)" \
  | python -m json.tool
```

Parametri per il comando **`endpoints`** per lo stato dell'endpoint dell'istanza

- **`id`** (obbligatorio): identificatore univoco dell'endpoint di inferenza.

Tipo: stringa.

- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Eliminazione di un endpoint dell'istanza con il comando **`endpoints`** in Neptune ML

Un comando `endpoints` di esempio in Neptune ML per l'eliminazione di un endpoint dell'istanza ha il seguente aspetto:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)"
```

Oppure questo:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)?
clean=true"
```

Parametri per il comando **`endpoints`** per l'eliminazione di un endpoint di inferenza

- **`id`** (obbligatorio): identificatore univoco dell'endpoint di inferenza.

  Tipo: stringa.

- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

- **`clean`** (facoltativo): indica che devono essere eliminati anche tutti gli artefatti relativi a questo endpoint.

  Tipo: Boolean Default: `FALSE`

Elenco degli endpoint di inferenza con il comando **`endpoints`** in Neptune ML

Un comando `endpoints` in Neptune ML per l'elenco degli endpoint di inferenza ha il seguente aspetto:

```
curl -s "https://(your Neptune endpoint)/ml/endpoints" \
  | python -m json.tool
```

Oppure questo:

```
curl -s "https://(your Neptune endpoint)/ml/endpoints?maxItems=3" \
  | python -m json.tool
```

Parametri per il comando **`dataprocessing`** per l'elenco degli endpoint di inferenza

- **`maxItems`** (facoltativo): numero massimo di elementi da restituire.

  Tipo: numero intero. Default: `10` Valore massimo consentito: `1024`.
- **`neptuneIamRoleArn`** (facoltativo): nome ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3.

  Tipo: stringa. Nota: deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Errori ed eccezioni dell'API di gestione Neptune ML

Tutte le eccezioni dell'API di gestione Neptune ML restituiscono un codice HTTP 400. Dopo aver ricevuto una di queste eccezioni, è consigliabile non ripetere il comando che ha generato l'eccezione.

- **`MissingParameterException`** – Messaggio di errore:

  ```
  Required credentials are missing. Please add IAM role to the cluster or pass as a parameter to this request.
  ```

- **`InvalidParameterException`** – Messaggi di errore:
  - `Invalid ML instance type.`
  - `Invalid ID provided. ID can be 1-48 alphanumeric characters.`
  - `Invalid ID provided. Must contain only letters, digits, or hyphens.`
  - `Invalid ID provided. Please check whether a resource with the given ID exists.`
  - `Another resource with same ID already exists. Please use a new ID.`
  - `Failed to stop the job because it has already completed or failed.`
- **`BadRequestException`** – Messaggi di errore:
  - `Invalid S3 URL or incorrect S3 permissions. Please check your S3 configuration.`
  - `Provided ModelTraining job has not completed.`
  - `Provided SageMaker Training job has not completed.`
  - `Provided MLDataProcessing job is not completed.`
  - `Provided MLModelTraining job doesn't exist.`
  - `Provided ModelTransformJob doesn't exist.`
  - `Unable to find SageMaker resource. Please check your input.`

Limiti di Neptune ML

- I tipi di inferenza attualmente supportati sono: classificazione dei nodi, regressione dei nodi, classificazione degli archi, regressione degli archi e la previsione dei collegamenti (consulta Funzionalità di Neptune ML).
- Le dimensioni massime del grafo supportate da Neptune ML dipendono dalla quantità di memoria e di spazio di archiviazione richiesti durante la preparazione dei dati, l'addestramento del modello e l'inferenza.
  - Le dimensioni massime della memoria di un'istanza di elaborazione dei dati SageMaker sono di 768 GB. Di conseguenza, la fase di elaborazione dei dati ha esito negativo se richiede più di 768 GB di memoria.
  - Le dimensioni massime della memoria di un'istanza di addestramento SageMaker sono di 732 GB. Di conseguenza, la fase di addestramento ha esito negativo se richiede più di 732 GB di memoria.
- Le dimensioni massime di un payload di inferenza per un endpoint SageMaker sono di 6 MiB. Di conseguenza, l'inferenza induttiva ha esito negativo se il payload del sottografo supera questo limite.
- Neptune ML è attualmente disponibile solo nelle regioni in cui Neptune e gli altri servizi da cui dipende (come AWS Lambda, il Gateway Amazon API e Amazon SageMaker) sono tutti supportati.

  Esistono differenze tra la regione Cina (Pechino) e la regione Cina (Ningxia) in merito all'utilizzo predefinito dell'autenticazione IAM, come spiegato qui, oltre ad altre differenze.
- Gli endpoint di inferenza per la previsione dei collegamenti avviati da Neptune ML al momento possono solo prevedere i collegamenti possibili con i nodi presenti nel grafo durante l'addestramento.

  Ad esempio, si consideri un grafo con i vertici `User` e `Movie` e gli archi `Rated`. Utilizzando un modello di raccomandazione di previsione dei collegamenti Neptune ML, è possibile aggiungere un nuovo utente al grafo affinché il modello preveda i film per tale utente, ma il modello può raccomandare solo i film presenti al momento dell'addestramento del modello. Sebbene l'incorporamento del nodo `User` venga calcolato in tempo reale utilizzando il relativo sottografo locale e il modello GNN e possa quindi cambiare nel tempo in base alle valutazioni dei film da parte degli utenti, per la raccomandazione finale viene confrontato con gli incorporamenti di film statici e precalcolati.
- I modelli di incorporamento del grafo della conoscenza supportati da Neptune ML funzionano solo per le attività di previsione dei collegamenti e le rappresentazioni sono specifiche per i vertici e

i tipi di archi presenti nel grafo durante l'addestramento. Ciò significa che tutti i vertici e i tipi di archi a cui viene fatto riferimento in una query di inferenza dovevano essere presenti nel grafo durante l'addestramento. Non è possibile eseguire previsioni per nuovi tipi di archi o vertici senza riaddestrare il modello.

Limitazioni delle risorse SageMaker

A seconda delle attività e dell'utilizzo delle risorse nel tempo, è possibile che vengano visualizzati messaggi di errore che indicano che è stata superata la quota (ResourceLimitExceeded) e che è necessario aumentare le risorse SageMaker. In tal caso, segui i passaggi della procedura Richiesta di un aumento delle quote dei servizi per le risorse SageMaker in questa pagina per richiedere un aumento della quota al Supporto AWS.

I nomi delle risorse SageMaker corrispondono alle fasi di Neptune ML come indicato di seguito:

- La risorsa `ProcessingJob` SageMaker viene usata dai processi di elaborazione dei dati, formazione dei modelli e trasformazione dei modelli in Neptune.
- La risorsa `HyperParameterTuningJob` SageMaker viene usata per i processo di addestramento dei modelli in Neptune.
- La risorsa `TrainingJob` SageMaker viene usata per i processo di addestramento dei modelli in Neptune.
- La risorsa `Endpoint` SageMaker viene usata dagli endpoint di inferenza in Neptune.

Monitoraggio delle risorse Amazon Neptune

Amazon Neptune supporta vari metodi per il monitoraggio delle prestazioni e dell'utilizzo del database:

- Stato dell'istanza: controlla lo stato del motore di database a grafo di un cluster Neptune, scopri quale versione del motore è installata e ottieni altre informazioni sull'istanza utilizzando l'API dello stato dell'istanza.
- API di riepilogo del grafo: l'API di riepilogo del grafo consente di ottenere rapidamente una comprensione di alto livello della dimensione e del contenuto dei dati del grafo.

> **Note**
>
> Poiché l'API di riepilogo del grafo si basa sulle statistiche DFE, è disponibile solo quando le statistiche sono abilitate, il che non è il caso dei tipi di istanza T3 e T4g.

- Amazon CloudWatch — Neptune invia automaticamente i parametri e supporta anche gli CloudWatch allarmi. CloudWatch Per ulteriori informazioni, consulta the section called “Usando CloudWatch”.
- File di log di audit: puoi visualizzare, scaricare o controllare i file di log del database tramite la console. Per ulteriori informazioni, consulta the section called “Log di audit con Neptune”.
- Pubblicazione dei log su Amazon CloudWatch Logs: puoi configurare un cluster Neptune DB per pubblicare i dati dei log di controllo in un gruppo di log in Amazon Logs. CloudWatch Con CloudWatch Logs, puoi eseguire analisi in tempo reale dei dati di log, utilizzarli CloudWatch per creare allarmi e visualizzare metriche e utilizzare CloudWatch Logs per archiviare i record di log in uno storage altamente durevole. Per informazioni, consulta Tronchi di Nettuno CloudWatch .
- AWS CloudTrail— Neptune supporta la registrazione delle API utilizzando. CloudTrail Per ulteriori informazioni, consulta the section called “Registrazione delle chiamate API Neptune con AWS CloudTrail”.
- Sottoscrizioni alle notifiche di eventi: effettua la sottoscrizione agli eventi di Neptune per rimanere informato su ciò che accade. Per ulteriori informazioni, consulta the section called “Notifiche di eventi”.
- Tagging: usa i tag per aggiungere metadati alle risorse Neptune e monitorare l'utilizzo in base a tag. Per ulteriori informazioni, consulta the section called “Applicazione di tag alle risorse Neptune”.

Argomenti

Controllo dello stato di un'istanza Neptune

Amazon Neptune fornisce un meccanismo per controllare lo stato del database a grafo sull'host. È anche un buon metodo per confermare che sei in grado di connetterti a un'istanza.

Per verificare lo stato di integrità di un'istanza e ottenere lo stato del cluster di database utilizzando `curl`:

```
curl -G https://your-neptune-endpoint:port/status
```

Oppure, a partire dal rilascio del motore 1.2.1.0.R6, puoi utilizzare il seguente comando della CLI:

```
aws neptunedata get-engine-status
```

Se l'istanza è integra, il comando `status` restituisce un oggetto JSON con i campi seguenti:

- **`status`**: impostato su `"healthy"` se l'istanza non presenta problemi.

  Se l'istanza è in fase di ripristino dopo un arresto anomalo o un riavvio e ci sono transazioni attive in esecuzione dall'ultimo arresto del server, `status` è impostato su `"recovery"`.
- **`startTime`**: impostato sull'ora UTC in cui è iniziato il processo del server corrente.
- **`dbEngineVersion`**: impostato sulla versione del motore Neptune in esecuzione sul cluster database.

Se è stata applicata una patch manualmente a questa versione del motore dopo il rilascio, il numero della versione ha il prefisso `"Patch-"`.

- **`role`**: impostato su `"reader"` se l'istanza è una replica di lettura o su `"writer"` se è l'istanza primaria.
- **`dfeQueryEngine`**: impostato su `"enabled"` se il motore DFE è completamente abilitato o su `viaQueryHint` se il motore DFE viene utilizzato solo con query per le quali l'hint di query `useDFE` è impostato su `true` (`viaQueryHint` è l'impostazione predefinita).
- **`gremlin`**: contiene informazioni sul linguaggio di query Gremlin disponibile nel cluster. In particolare, contiene un `version` campo che specifica la TinkerPop versione corrente utilizzata dal motore.
- **`sparql`**: contiene informazioni sul linguaggio di query SPARQL disponibile nel cluster. In particolare, contiene un campo `version` che specifica la versione corrente di SPARQL utilizzata dal motore.
- **`opencypher`**: contiene informazioni sul linguaggio di query openCypher disponibile nel cluster. In particolare, contiene un campo `version` che specifica la versione corrente di operCypher utilizzata dal motore.
- **`labMode`**: contiene le impostazioni della Modalità di laboratorio utilizzate dal motore.
- **`rollingBackTrxCount`**: se sono presenti transazioni di cui è stato eseguito il rollback, questo campo è impostato sul numero di tali transazioni. Se non ne esistono, il campo non viene visualizzato.
- **`rollingBackTrxEarliestStartTime`**: impostato sull'ora di inizio del rollback della prima transazione. Se non è stato eseguito il rollback di alcuna transazione, il campo non viene visualizzato.
- **`features`**: contiene informazioni sullo stato delle funzionalità abilitate nel cluster database.
  - **`lookupCache`**: stato corrente della Cache di ricerca. Questo campo viene visualizzato solo per tipi di istanza `R5d`, poiché sono le uniche istanze in cui può esistere una cache di ricerca. Il campo è un oggetto JSON nel formato:

    ```
    "lookupCache": {
      "status": "current lookup cache status"
    }
    ```

    Per un'istanza `R5d`:

    - Se la cache di ricerca è abilitata, lo stato è riportato come `"Available"`.

- Se la cache di ricerca è stata disabilitata, lo stato è riportato come `"Disabled"`.
- Se è stato raggiunto il limite del disco per l'istanza, lo stato viene riportato come `"Read Only Mode - Storage Limit Reached"`.

- **`ResultCache`**: stato corrente della Memorizzazione nella cache dei risultati delle query. Questo campo è un oggetto JSON nel formato:

```
"ResultCache": {
  "status": "current results cache status"
}
```

  - Se la cache dei risultati è stata abilitata, lo stato è riportato come `"Available"`.
  - Se la cache è disabilitata, lo stato è riportato come `"Disabled"`.
- **`IAMAuthentication`**— Speciifica se l'autenticazione AWS Identity and Access Management (IAM) è stata abilitata o meno sul cluster DB:
  - Se l'autenticazione IAM è stata abilitata, lo stato è riportato come `"enabled"`.
  - Se l'autenticazione IAM è disabilitata, lo stato è riportato come `"disabled"`.
- **`Streams`**: specifica se la funzionalità Neptune Streams è stata abilitata o meno sul cluster database:
  - Se i flussi sono abilitati, lo stato è riportato come `"enabled"`.
  - Se i flussi sono disabilitati, lo stato è riportato come `"disabled"`.
- **`AuditLog`**: uguale a `enabled` se i log di audit sono abilitati, in caso contrario `disabled`.
- **`SlowQueryLogs`**: uguale a `info` o `debug` se la registrazione di log delle query lente è abilitata, in caso contrario `disabled`.
- **`QueryTimeout`**: valore, in millisecondi, del timeout delle query.

- **`settings`**: impostazioni applicate all'istanza:
  - **`clusterQueryTimeoutInMs`**: valore, in millisecondi, del timeout delle query, impostato per l'intero cluster.
  - **`SlowQueryLogsThreshold`**: valore, in millisecondi, del timeout delle query, impostato per l'intero cluster.
- **`serverlessConfiguration`**: impostazioni serverless per un cluster se è in esecuzione come serverless:
  - **`minCapacity`**: dimensioni minime fino alle quali può ridursi un'istanza serverless nel cluster database, in unità di capacità Neptune (NCU, Neptune Capacity Unit).

- **`maxCapacity`**: dimensioni massime fino alle quali può aumentare un'istanza serverless nel cluster database, in unità di capacità Neptune (NCU, Neptune Capacity Unit).

Esempio di output del comando di stato dell'istanza

Di seguito è riportato un esempio dell'output del comando di stato dell'istanza, (in questo caso, eseguito su un'istanza R5d):

```
{
  'status': 'healthy',
  'startTime': 'Thu Aug 24 21:47:12 UTC 2023',
  'dbEngineVersion': '1.2.1.0.R4',
  'role': 'writer',
  'dfeQueryEngine': 'viaQueryHint',
  'gremlin': {'version': 'tinkerpop-3.6.2'},
  'sparql': {'version': 'sparql-1.1'},
  'opencypher': {'version': 'Neptune-9.0.20190305-1.0'},
  'labMode': {
    'ObjectIndex': 'disabled',
    'ReadWriteConflictDetection': 'enabled'
  },
  'features': {
    'SlowQueryLogs': 'disabled',
    'ResultCache': {'status': 'disabled'},
    'IAMAuthentication': 'disabled',
    'Streams': 'disabled',
    'AuditLog': 'disabled'
  },
  'settings': {
    'clusterQueryTimeoutInMs': '120000',
    'SlowQueryLogsThreshold': '5000'
  },
  'serverlessConfiguration': {
    'minCapacity': '1.0',
    'maxCapacity': '128.0'
  }
}
```

Se si verifica un problema con l'istanza, il comando di stato restituisce il codice di errore `HTTP 500`. Se l'host non è raggiungibile, la richiesta scade. Verifica di accedere all'istanza dal cloud privato virtuale (VPC, Virtual Private Cloud) e che i gruppi di sicurezza consentano l'accesso.

Monitoraggio di Neptune tramite Amazon CloudWatch

Amazon Neptune e CloudWatch Amazon sono integrati in modo da poter raccogliere e analizzare i parametri delle prestazioni. Puoi monitorare questi parametri utilizzando la CloudWatch console, AWS Command Line Interface (AWS CLI) o l'API. CloudWatch

CloudWatch consente inoltre di impostare allarmi in modo da poter essere avvisati se un valore metrico supera una soglia specificata. Puoi anche impostare CloudWatch Events per intraprendere azioni correttive in caso di violazione. Per ulteriori informazioni sull'utilizzo CloudWatch e sugli allarmi, consulta la documentazione. CloudWatch

Argomenti

Visualizzazione CloudWatch dei dati (console)

Per visualizzare CloudWatch i dati per un cluster Neptune (console)

1. Accedi AWS Management Console e apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Nel riquadro di navigazione, seleziona Parametri.
3. Nel riquadro Tutte le metriche, scegli Neptune, quindi scegli DB. ClusterIdentifier
4. Nel riquadro superiore, scorri verso il basso per visualizzare l'elenco completo dei parametri per il cluster. Le opzioni delle metriche Neptune disponibili vengono visualizzate nell'elenco Visualizzazione.

Per selezionare o deselezionare un singolo parametro, nel riquadro risultati seleziona la casella di controllo accanto al nome della risorsa e al parametro. I grafici che mostrano i parametri per gli elementi selezionati vengono visualizzati nella parte inferiore della console. Per ulteriori informazioni sui CloudWatch grafici, consulta Graph Metrics nella Amazon CloudWatch User Guide.

Visualizzazione dei CloudWatch dati ()AWS CLI

Per visualizzare CloudWatch i dati per un cluster Neptune ()AWS CLI

1. Installa il. AWS CLI Per istruzioni, consulta la Guida per l'utente di AWS Command Line Interface.
2. Usa il AWS CLI per recuperare informazioni. I CloudWatch parametri rilevanti per Neptune sono elencati in. Metriche di Neptune CloudWatch

   L'esempio seguente recupera le CloudWatch metriche per il numero di richieste Gremlin al secondo per il cluster. `gremlin-cluster`

   ```
   aws cloudwatch get-metric-statistics \
       --namespace AWS/Neptune  --metric-name GremlinRequestsPerSec \
       --dimensions Name=DBClusterIdentifier,Value=gremlin-cluster \
       --start-time 2018-03-03T00:00:00Z --end-time 2018-03-04T00:00:00Z \
       --period 60 --statistics=Average
   ```

Visualizzazione dei dati (API) CloudWatch

CloudWatch supporta anche un'`Query`azione che consente di richiedere informazioni a livello di codice. Per ulteriori informazioni, consulta la documentazione dell'API CloudWatch Query e Amazon CloudWatch API Reference.

Quando un' CloudWatch azione richiede un parametro specifico per il monitoraggio di Neptune, ad esempio, utilizzare i `MetricName` valori elencati in. Metriche di Neptune CloudWatch

L'esempio seguente mostra una CloudWatch richiesta di basso livello, utilizzando i seguenti parametri:

- `Statistics.member.1` = `Average`
- `Dimensions.member.1` = `DBClusterIdentifier=gremlin-cluster`
- `Namespace` = `AWS/Neptune`
- `StartTime` = `2013-11-14T00:00:00Z`
- `EndTime` = `2013-11-16T00:00:00Z`
- `Period` = `60`

- `MetricName = GremlinRequestsPerSec`

Ecco come si presenta la CloudWatch richiesta. Tuttavia, questo esempio ha solo lo scopo di mostrare il formato della richiesta; dovrai crearne una personalizzata in base ai parametri e all'intervallo temporale da te definiti.

```
https://monitoring.amazonaws.com/
      ?SignatureVersion=2
      &Action=GremlinRequestsPerSec
      &Version=2010-08-01
      &StartTime=2018-03-03T00:00:00
      &EndTime=2018-03-04T00:00:00
      &Period=60
      &Statistics.member.1=Average
      &Dimensions.member.1=DBClusterIdentifier=gremlin-cluster
      &Namespace=AWS/Neptune
      &MetricName=GremlinRequests
      &Timestamp=2018-03-04T17%3A48%3A21.746Z
      &AWSAccessKeyId=AWS Access Key ID;
      &Signature=signature
```

Utilizzo CloudWatch per monitorare le prestazioni delle istanze DB in Neptune

Puoi utilizzare le CloudWatch metriche in Neptune per monitorare ciò che accade sulle tue istanze DB e tenere traccia della lunghezza della coda di query osservata dal database. Le metriche riportate di seguito sono particolarmente utili:

- **`CPUUtilization`**: mostra la percentuale di utilizzo della CPU.
- **`VolumeWriteIOPs`**: mostra il numero medio delle operazioni I/O di scrittura sul disco nel volume del cluster, indicato a intervalli di 5 minuti.
- **`MainRequestQueuePendingRequests`**: mostra il numero di richieste in attesa di esecuzione nella coda di input.

È inoltre possibile scoprire quante richieste sono in sospeso sul server utilizzando l'endpoint di stato delle query Gremlin con il parametro `includeWaiting`. In questo modo si ottiene lo stato di tutte le query in attesa.

I seguenti indicatori possono consentono di modificare le strategie di provisioning e di query di Neptune per migliorare l'efficienza e le prestazioni:

- Latenza costante, valore `CPUUtilization` alto, valore `VolumeWriteIOPs` alto e valore `MainRequestQueuePendingRequests` basso insieme indicano che il server è attivamente impegnato nell'elaborazione di richieste di scrittura simultanee a un ritmo sostenibile, con tempi di attesa I/O minimi.
- Latenza costante, valore `CPUUtilization` basso, valore `VolumeWriteIOPs` basso e valore `MainRequestQueuePendingRequests` pari a zero insieme indicano una capacità eccessiva sull'istanza database primaria per l'elaborazione delle richieste di scrittura.
- Valore `CPUUtilization` alto e valore `VolumeWriteIOPs` alto ma latenza variabile e valore `MainRequestQueuePendingRequests` insieme indicano che si sta inviando più lavoro di quanto il server sia in grado di elaborare in un determinato intervallo. Prendere in considerazione la possibilità di creare o ridimensionare le richieste batch in modo da svolgere la stessa quantità di lavoro con meno sovraccarico transazionale e/o di dimensionare l'istanza primaria per aumentare il numero di thread di query in grado di elaborare contemporaneamente le richieste di scrittura.
- Valore `CPUUtilization` basso con valore `VolumeWriteIOPs` alto significa che i thread di query sono in attesa del completamento delle operazioni di I/O al livello di archiviazione. Se si notano latenze variabili e un certo aumento del valore `MainRequestQueuePendingRequests`, prendere in considerazione la possibilità di creare o ridimensionare le richieste batch in modo da svolgere la stessa quantità di lavoro con un minore sovraccarico transazionale.

Metriche di Neptune CloudWatch

> **Note**
>
> Amazon Neptune invia i parametri solo quando hanno un CloudWatch valore diverso da zero. Per tutti le metriche Neptune, la granularità dell'aggregazione è di 5 minuti.

Argomenti

Metriche di Neptune CloudWatch

La tabella seguente elenca le CloudWatch metriche supportate da Neptune.

> **Note**
>
> Tutti le metriche cumulative vengono azzerate ogni volta che il server viene riavviato, sia per manutenzione, riavvio o ripristino dopo un arresto anomalo.

Metriche di Neptune CloudWatch

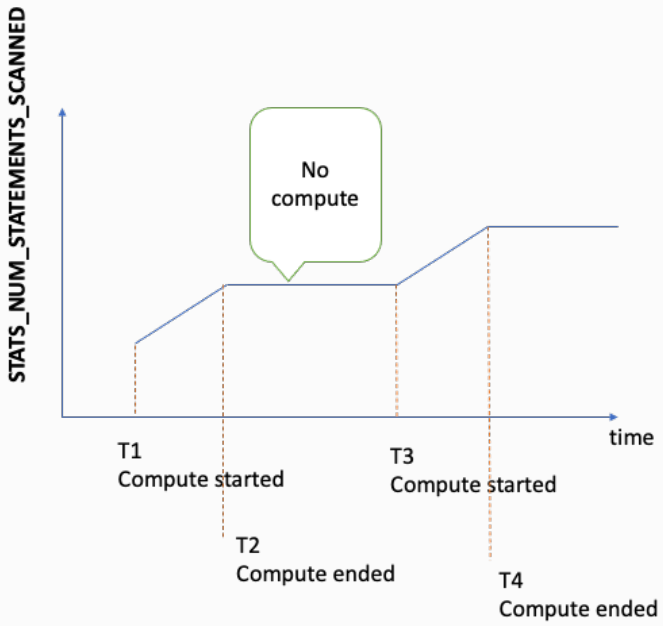
Parametro	Descrizione
`BackupRetentionPeriodStorageUsed`	La quantità totale di spazio di archiviazione di backup, espressa in byte, utilizzata per il supporto dalla finestra di conservazione del backup del cluster database Neptune. Incluso nel totale riportato dal parametro `TotalBackupStorageBilled` .
`BufferCacheHitRatio`	La percentuale di richieste gestite dalla cache del buffer. Questa metrica può essere utile per diagnosticare la latenza delle query, poiché mancati riscontri della cache provocano una latenza significativa. Se il rapporto di riscontri della cache è inferiore al 99,9%, valutare la possibilità di aggiornare il tipo di istanza per memorizzare nella cache più dati in memoria.
`ClusterReplicaLag`	Per una replica di lettura, il ritardo durante la replica degli aggiornamenti dall'istanza principale, in millisecondi.
`ClusterReplicaLagMaximum`	Il ritardo massimo tra l'istanza primaria e ogni istanza database Neptune nel cluster database, in millisecondi.

Parametro	Descrizione
`ClusterReplicaLagMinimum`	Il ritardo minimo tra l'istanza primaria e ogni istanza database Neptune nel cluster database, in millisecondi.
`CPUUtilization`	La percentuale di utilizzo della CPU.
`EngineUptime`	Il periodo di esecuzione dell'istanza, in secondi.
`FreeableMemory`	La quantità di memoria RAM disponibile, in byte.
`GlobalDbDataTransferBytes`	Il numero di byte di dati di redo log trasferiti dal primario Regione AWS a uno secondari o Regione AWS in un database globale di Neptune.
`GlobalDbReplicatedWriteIO`	Numero di operazioni di I/O di scrittura replicate dalla Regione AWS primaria nel database globale al volume cluster in una Regione AWS secondaria. I calcoli di fatturazione per ogni cluster database in un database globale Neptune utilizzano la metrica `VolumeWriteIOPS` per determinare il numero di scritture eseguite all'interno del cluster. Per il cluster database primario, i calcoli di fatturazione usano `GlobalDbReplicatedWriteIO` per tenere conto della replica tra regioni nei cluster database secondari.
`GlobalDbProgressLag`	Numero di millisecondi di ritardo del cluster secondario rispetto al cluster primario sia per le transazioni utente che per le transazioni di sistema.

Parametro	Descrizione
`GremlinRequestsPerSec`	Numero di richieste al secondo al motore Gremlin.
`GremlinWebSocketOpenConnections`	Il numero di WebSocket connessioni aperte verso Neptune.
`LoaderRequestsPerSec`	Numero di richieste dello strumento di caricamento al secondo.
`MainRequestQueuePendingRequests`	Numero di richieste in attesa di esecuzione nella coda di input. Neptune inizia a limitare le richieste quando superano la capacità massima della coda.
`NCUUtilization`	Applicabile solo a un'istanza database o un cluster DB Neptune Serverless. A livello di istanza, riporta una percentuale calcolata come il numero di unità di capacità Neptune (NCU) attualmente utilizzate dall'istanza in questione , diviso per l'impostazione della capacità NCU massima per il cluster. Un'unità di capacità Neptune (NCU, Neptune Capacity Unit) è costituita da 2 GiB (gibibyte) di memoria (RAM) insieme alle reti e alla capacità del processore virtuale (vCPU) associate. A livello di cluster, `NCUUtilization` riporta la percentuale di capacità massima utilizzata dal cluster nel suo complesso.
`NetworkThroughput`	La velocità di trasmissione effettiva della rete in byte al secondo in entrata e in uscita dei client per ogni istanza del cluster database Neptune. Questa velocità di trasmissione effettiva non include il traffico di rete tra le istanze del cluster database e il volume del cluster.

Parametro	Descrizione
NetworkTransmitThroughput	La velocità di trasmissione effettiva della rete in byte al secondo in uscita dei client per ogni istanza del cluster database Neptune. Questa velocità di trasmissione effettiva non include il traffico di rete tra le istanze del cluster database e il volume del cluster.
NumTxCommitted	Il numero di transazioni impegnate correttamente al secondo.
NumTxOpened	Il numero di transazioni aperte sul server al secondo.
NumTxRolledBack	Per le query di scrittura, il numero di transazioni al secondo di cui è stato eseguito il rollback sul server a causa di errori. Per le query di sola lettura, questa metrica è uguale al numero di transazioni di sola lettura completate al secondo.
OpenCypherRequestsPerSec	Numero di richieste al secondo (sia HTTPS che Bolt) al motore openCypher.
OpenCypherBoltOpenConnections	Il numero di connessioni Bolt aperte a Neptune.

Parametro	Descrizione
`ServerlessDatabaseCapacity`	Come parametro a livello di istanza, `ServerlessDatabaseCapacity` riporta la capacità corrente di una determinata istanza serverless Neptune, espressa in NCU. Un'unità di capacità Neptune (NCU, Neptune Capacity Unit) è costituita da 2 GiB (gibibyte) di memoria (RAM) insieme alle reti e alla capacità del processore virtuale (vCPU) associate. A livello di cluster, `ServerlessDatabaseCapacity` riporta la media di tutti i valori `ServerlessDatabaseCapacity` delle istanze database del cluster.
`SnapshotStorageUsed`	La quantità totale di spazio di archiviazione di backup, espressa in byte, utilizzata da tutti gli snapshot per un cluster database Neptune al di fuori della finestra di conservazione dei backup. Incluso nel totale riportato dal parametro `TotalBackupStorageBilled` .
`SparqlRequestsPerSec`	Il numero di richieste al secondo al motore SPARQL.

Parametro	Descrizione
`StatsNumStatementsScanned`	Numero totale di istruzioni analizzate per le statistiche DFE dall'avvio del server. Ogni volta che viene attivato il calcolo delle statistiche, questo numero aumenta ma quando non viene eseguito alcun calcolo, rimane statico. Di conseguenza, se lo si rappresenta graficamente nel tempo, è possibile capire quando il calcolo è avvenuto e quando non è avvenuto:  Osservando la pendenza del grafico nei periodi in cui la metrica è in aumento, si può anche capire la velocità di calcolo. Se non esiste una metrica di questo tipo, significa che la funzionalità delle statistiche è disabilitata nel cluster database o che la versione del motore in uso non la include. Se il valore della metrica è zero, significa che non è stato effettuato alcun calcolo delle statistiche.

Parametro	Descrizione
`TotalBackupStorageBilled`	La quantità totale di spazio di archiviazione di backup fatturata per un determinato cluster database Neptune, in byte. Include lo storage di backup misurato dai parametri `BackupRetentionPeriodStorageUsed` e `SnapshotStorageUsed` .
`TotalRequestsPerSec`	Il numero totale di richieste al secondo al server da tutte le origini.
`TotalClientErrorsPerSec`	Il numero totale al secondo di richieste che hanno causato errori a causa di problemi lato client.
`TotalServerErrorsPerSec`	Il numero totale al secondo di richieste che hanno causato errori nel server a causa di errori interni.

Parametro	Descrizione
`UndoLogListSize`	Il numero di log di annullamento nell'elenco dei log di annullamento. I log di annullamento contengono i record delle transazioni sottoposte a commit che scadono quando tutte le transazioni attive sono più recenti dell'ora del commit. I record scaduti vengono eliminati periodicamente. L'elimina zione dei record per le operazioni di eliminazi one può richiedere più tempo rispetto ai record per altri tipi di transazione. L'eliminazione viene eseguita esclusivamente dall'istanza di scrittura del cluster database, quindi la velocità di eliminazione dipende dal tipo di istanza di scrittura. Se il valore `UndoLogListSize` è elevato e in crescita nel cluster database, aggiornare l'istanza di scrittura per aumentare la velocità di eliminazi one. Inoltre, se si sta effettuando l'aggiornamento alla versione `1.2.0.0` o superiore del motore da una versione precedente alla `1.2.0.0`, assicurarsi innanzitutto che il valore `UndoLogListSize` sia vicino a 0. Poiché le versioni `1.2.0.0` e successive del motore utilizzano un formato diverso per i log di annullamento, l'aggiornamento può iniziare solo dopo che i log di annullamento precedenti sono stati completamente eliminati. Per ulteriori informazioni, consulta Aggiornamento alla versione 1.2.0.0 o successiva.

Parametro	Descrizione
`VolumeBytesUsed`	La quantità totale di archiviazione allocata al cluster database Neptune, espressa in byte. Questa è la quantità di spazio di archiviazione che ti viene fatturata. È la quantità massima di archiviazione allocata al cluster DB in qualsiasi momento della sua esistenza, non la quantità che si sta attualmente utilizzando (consulta Fatturazione dell'archiviazione Neptune).
`VolumeReadIOPs`	Numero medio delle operazioni I/O di lettura fatturate da un volume di cluster, indicato a intervalli di 5 minuti. Le operazioni di lettura fatturate sono calcolate a livello del volume del cluster, aggregate da tutte le istanze nel cluster database Neptune e, in seguito, indicate a intervalli di 5 minuti.
`VolumeWriteIOPs`	Numero medio delle operazioni I/O di scrittura sul disco nel volume del cluster, indicato a intervalli di 5 minuti.

CloudWatch Metriche che ora sono obsolete in Neptune

L'uso di queste metriche Neptune è ora obsoleto. Sono ancora supportati, ma potrebbero essere eliminati in futuro man mano che nuove e migliori metriche diventano disponibili.

Parametro	Descrizione
`GremlinHttp1xx`	Numero di risposte HTTP 1xx per l'endpoint Gremlin al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http1xx`.

Parametro	Descrizione
`GremlinHttp2xx`	Numero di risposte HTTP 2xx per l'endpoint Gremlin al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http2xx`.
`GremlinHttp4xx`	Numero di errori HTTP 4xx per l'endpoint Gremlin al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http4xx`.
`GremlinHttp5xx`	Numero di errori HTTP 5xx per l'endpoint Gremlin al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http5xx`.
`GremlinErrors`	Numero di errori negli attraversamenti Gremlin.
`GremlinRequests`	Numero di richieste al motore Gremlin.
`GremlinWebSocketSuccess`	Numero di WebSocket connessioni riuscite all'endpoint Gremlin al secondo.
`GremlinWebSocketClientErrors`	Numero di errori del WebSocket client sull'endpoint Gremlin al secondo.
`GremlinWebSocketServerErrors`	Numero di errori del WebSocket server sull'endpoint Gremlin al secondo.
`GremlinWebSocketAvailableConnections`	Numero di potenziali WebSocket connessioni attualmente disponibili.

Parametro	Descrizione
`Http100`	Numero di risposte HTTP 100 per l'endpoint al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http1xx`.
`Http101`	Numero di risposte HTTP 101 per l'endpoint al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http1xx`.
`Http1xx`	Numero di risposte HTTP 1xx per l'endpoint al secondo.
`Http200`	Numero di risposte HTTP 200 per l'endpoint al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http2xx`.
`Http2xx`	Numero di risposte HTTP 2xx per l'endpoint al secondo.
`Http400`	Numero di errori HTTP 400 per l'endpoint al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http4xx`.
`Http403`	Numero di errori HTTP 403 per l'endpoint al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http4xx`.

Parametro	Descrizione
`Http405`	Numero di errori HTTP 405 per l'endpoint al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http4xx`.
`Http413`	Numero di errori HTTP 413 per l'endpoint al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http4xx`.
`Http429`	Numero di errori HTTP 429 per l'endpoint al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http4xx`.
`Http4xx`	Numero di errori HTTP 4xx per l'endpoint al secondo.
`Http500`	Numero di errori HTTP 500 per l'endpoint al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http5xx`.
`Http501`	Numero di errori HTTP 501 per l'endpoint al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http5xx`.
`Http5xx`	Numero di errori HTTP 5xx per l'endpoint al secondo.
`LoaderErrors`	Numero di errori delle richieste dello strumento di caricamento.

Parametro	Descrizione
`LoaderRequests`	Numero di richieste dello strumento di caricamento.
`SparqlHttp1xx`	Numero di risposte HTTP 1xx per l'endpoint SPARQL al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http1xx`.
`SparqlHttp2xx`	Numero di risposte HTTP 2xx per l'endpoint SPARQL al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http2xx`.
`SparqlHttp4xx`	Numero di errori HTTP 4xx per l'endpoint SPARQL al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http4xx`.
`SparqlHttp5xx`	Numero di errori HTTP 5xx per l'endpoint SPARQL al secondo. Al suo posto, ti consigliamo di utilizzare il nuovo parametro combinato `Http5xx`.
`SparqlErrors`	Numero di errori nelle query SPARQL.
`SparqlRequests`	Numero di richieste al motore SPARQL.
`StatusErrors`	Numero di errori dell'endpoint dello stato.
`StatusRequests`	Numero di richieste all'endpoint dello stato.

Dimensioni di Nettuno CloudWatch

Le metriche di Amazon Neptune vengono qualificate mediante i valori dell'account, il nome del grafo o l'operazione. Puoi utilizzare la CloudWatch console Amazon per recuperare i dati di Neptune insieme a qualsiasi dimensione nella tabella seguente.

Dimensione	Descrizione
`DBInstanceIdentifier`	Filtra i dati richiesti solo per un'istanza database specifica di un cluster.
`DBClusterIdentifier`	Filtra i dati richiesti per un cluster database Neptune specifico.
`DBClusterIdentifier` , `EngineName`	Filtra i dati in base al cluster. Il nome del motore per tutte le istanze Neptune è `neptune`.
`DBClusterIdentifier` , `Role`	Filtra i dati richiesti per un cluster database Neptune specifico, aggregando la metrica in base al ruolo dell'istanza (LETTURA/ SCRITTURA). Ad esempio, puoi aggregare i parametri per tutte le istanze READER che appartengono a un cluster.
`DBClusterIdentifier` , `SourceRegion`	Filtra i dati in base al cluster primario in una regione primaria del database globale.
`DatabaseClass`	Filtra i dati richiesti per tutte le istanze in una classe di database. Ad esempio, puoi aggregare i parametri per tutte le istanze che appartengono alla classe di database db.r4.large
`EngineName`	Il nome del motore per tutte le istanze Neptune è `neptune`.
`GlobalDbDBClusterIdentifier` , `SecondaryRegion`	Filtra i dati in base al cluster secondario di un database globale specificato in una regione secondaria.

Utilizzo dei log di audit con i cluster Amazon Neptune

Per effettuare l'audit dell'attività del cluster database Amazon Neptune, abilitare la raccolta dei log di audit impostando un parametro del cluster database. Una volta abilitati, i registri di controllo possono essere utilizzati per registrare una qualsiasi combinazione di eventi supportati. Puoi visualizzare o scaricare i log di audit per esaminarli.

Abilitazione dei log di audit Neptune

Utilizza il parametro `neptune_enable_audit_log` per abilitare (`1`) o disabilitare (`0`) i registri di controllo.

Imposta questo parametro nel gruppo di parametri che viene utilizzato dal cluster database. Puoi utilizzare la procedura mostrata in Modifica di un gruppo di parametri del cluster di database o di un gruppo di parametri di database per modificare il parametro utilizzando o utilizzare il AWS Management Console comando modify-db-cluster-parameter-group o il AWS CLI comando ClusterParameterGroup API ModifyDB per modificare il parametro a livello di codice.

Dopo aver modificato questo parametro, è necessario riavviare il cluster database per applicare la modifica.

Visualizzazione dei log di audit Neptune mediante la console

Puoi visualizzare e scaricare i registri di controllo utilizzando la AWS Management Console. Nella pagina Instances (Istanze), scegli l'istanza database per visualizzarne i dettagli, successivamente scorri verso la sezione Logs (Registri).

Per scaricare un file di log, seleziona il file nella sezione Logs (Registri) e successivamente Download (Scarica).

Dettagli del log di audit Neptune

I file di log sono in formato UTF-8. I registri vengono scritti in più file, il cui numero varia a seconda della dimensione dell'istanza. Per visualizzare gli ultimi eventi, potrebbe essere necessario esaminare tutti i file dei registri di controllo.

Le voci dei log non sono in ordine sequenziale. Per ordinarle, puoi utilizzare il valore `timestamp`.

I file di log sono ruotati quando raggiungono la dimensione di 100 MB in forma aggregata. Questo limite non è configurabile.

I file dei log di audit includono le seguenti informazioni delimitate da virgola, in righe, nell'ordine seguente:

Campo	Descrizione
Timestamp	Il timestamp Unix per l'evento registrato con una precisione al microsecondo.
ClientHost	Il nome host o l'IP da cui si connette l'utente.
ServerHost	Il nome host o l'IP dell'istanza per cui viene registrato l'evento.
ConnectionType	Il tipo di connessione. Può essere `Websocket` , `HTTP_POST` , `HTTP_GET` o `Bolt`.

ARN IAM del chiamante	L'ARN dell'utente IAM o del ruolo IAM utilizzato per firmare la richiesta. Vuoto se l'autenticazione IAM è disabilitata. Il formato è:
	`arn:`*`partition`*` :`*`service`*`:`*`region`*`:`*`account`*`:`*`resource`*
	Per esempio:
	`arn:aws:iam::123456789012:user/Anna`
	`arn:aws:sts::123456789012:assumed-role/AWSNeptuneNotebookRole/SageMaker`

Auth Context	Contiene un oggetto JSON serializzato contenente informazioni di autenticazione. Il campo `authenticationSucceeded` è `True` se l'utente è stato autenticato. Vuoto se l'autenticazione IAM è disabilitata.
HttpHeader	Le informazioni di intestazione HTTP. Può contenere una query. Connessioni vuote per WebSocket e Bolt.
Payload	La query Gremlin, SPARQL o openCypher.

Pubblicazione dei log di Neptune su Amazon Logs CloudWatch

Puoi configurare un cluster Neptune DB per pubblicare dati di log di audit e/o dati di log con query lente su un gruppo di log in Amazon Logs. CloudWatch Con CloudWatch Logs, puoi eseguire analisi

in tempo reale dei dati di log e utilizzarli CloudWatch per creare allarmi e visualizzare metriche. È possibile utilizzare CloudWatch Logs per archiviare i record di registro in un archivio altamente durevole.

Per pubblicare i log di controllo su CloudWatch Logs, i log di controllo devono essere abilitati in modo esplicito (vedi). Abilitazione dei registri di controllo Analogamente, per pubblicare i log con query lente su Logs, i log con query lente devono essere CloudWatch abilitati in modo esplicito (vedi). Utilizzo della registrazione di log delle query lente di Amazon Neptune

> **Note**
>
> Ricorda quanto segue:
>
> - Si applicano costi aggiuntivi quando si pubblicano i log su. CloudWatch Consulta la pagina CloudWatch dei prezzi per i dettagli.
> - Non puoi pubblicare log in CloudWatch Logs for the China (Pechino) o China (Ningxia).
> - Se l'esportazione dei dati del log è disabilitata, Neptune non elimina i gruppi di log o i flussi di log esistenti. Se l'esportazione dei dati di registro è disabilitata, i dati di registro esistenti rimangono disponibili nei CloudWatch registri, a seconda della conservazione dei log, e all'utente vengono comunque addebitati costi per i dati dei log di controllo archiviati. È possibile eliminare i flussi di log e i gruppi di log utilizzando la console CloudWatch Logs, o l' AWS CLI API Logs. CloudWatch

Utilizzo della console per pubblicare i log di Neptune nei registri CloudWatch

Per pubblicare i log di Neptune su Logs CloudWatch dalla console

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Scegliere il cluster database Neptune per cui pubblicare i dati dei log.
4. Per Operazioni, scegli Modifica.
5. Nella sezione Esportazioni dei log, scegli i log che desideri iniziare a pubblicare su Logs. CloudWatch

6. Scegliere Continue (Continua) e quindi selezionare Modify DB Cluster (Modifica cluster DB) nella pagina di riepilogo.

Utilizzo della CLI per pubblicare i log di controllo di Neptune su Logs CloudWatch

È possibile creare un nuovo cluster DB che pubblica i log di controllo in CloudWatch Logs utilizzando il AWS CLI `create-db-cluster` comando con i seguenti parametri:

```
aws neptune create-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["audit"]'
```

È possibile configurare un cluster DB esistente per pubblicare i log di controllo nei CloudWatch registri utilizzando il AWS CLI `modify-db-cluster` comando con i seguenti parametri:

```
aws neptune modify-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Utilizzo della CLI per pubblicare i log di Neptune con query lente su Logs CloudWatch

È inoltre possibile creare un nuovo cluster DB che pubblica i log con query lente su Logs utilizzando il comando con i CloudWatch seguenti parametri: AWS CLI `create-db-cluster`

```
aws neptune create-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["slowquery"]'
```

Allo stesso modo, è possibile configurare un cluster DB esistente per pubblicare log con query lente su Logs utilizzando il comando con i seguenti CloudWatch parametri: AWS CLI `modify-db-cluster`

```
aws neptune modify-db-cluster --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["slowquery"]}'
```

Monitoraggio degli eventi di Neptune Log in Amazon CloudWatch

Dopo aver abilitato i log di Neptune, puoi monitorare gli eventi di registro in Amazon Logs. CloudWatch Un nuovo gruppo di log viene creato automaticamente per il cluster database Neptune nel seguente prefisso, in cui *`cluster-name`* rappresenta il nome del cluster database e *`log_type`* rappresenta il tipo di log:

```
/aws/neptune/cluster-name/log_type
```

Ad esempio, se configuri la funzione di esportazione per includere il log di audit per un cluster di database denominato `mydbcluster`, i dati di log vengono archiviati nel gruppo di log `/aws/neptune/mydbcluster/audit`.

Tutti gli eventi di tutte le istanze database in un cluster DB vengono inviati a un gruppo di log utilizzando flussi di log diversi.

Se esiste un gruppo di log con il nome specificato, Neptune utilizza quel gruppo di log per esportare i dati di log per il cluster database Neptune. Puoi utilizzare la configurazione automatizzata, ad esempio per creare gruppi di log con periodi di conservazione dei log predefiniti, filtri metrici e accesso dei clienti. AWS CloudFormation In caso contrario, viene creato automaticamente un nuovo gruppo di log utilizzando il periodo di conservazione dei log predefinito, Never Expire, in Logs. CloudWatch

È possibile utilizzare la console CloudWatch Logs, l'API Logs o l' AWS CLI API CloudWatch Logs per modificare il periodo di conservazione dei log. Per ulteriori informazioni sulla modifica dei periodi di conservazione dei log in CloudWatch Logs, consulta Change Log Data Retention in Logs. CloudWatch

È possibile utilizzare la console CloudWatch Logs AWS CLI, l'API Logs o l'API CloudWatch Logs per cercare informazioni all'interno degli eventi di registro per un cluster DB. Per ulteriori informazioni sulla ricerca e l'applicazione di filtri per i dati di log, consulta Ricerca e filtraggio dei dati di log.

Abilitazione di Amazon CloudWatch Logs per un notebook Neptune

CloudWatch I registri per i notebook Neptune sono disabilitati per impostazione predefinita. Seguire questi passaggi per abilitarlo, per il debug o per altri scopi:

Utilizzo di AWS Management Console per abilitare CloudWatch i log per un notebook Neptune

1. Apri la SageMaker console Amazon all'indirizzo https://console.aws.amazon.com/sagemaker/.
2. Nel riquadro di navigazione a sinistra scegli Notebook, quindi Istanze notebook. Cerca il nome del notebook Neptune per il quale desideri abilitare i log.
3. Vai alla pagina dei dettagli scegliendo il nome dell'istanza notebook menzionata nel passaggio precedente.
4. Se l'istanza notebook è in esecuzione, seleziona il pulsante Arresta in alto a destra nella pagina dei dettagli del notebook.
5. In Autorizzazioni e crittografia è disponibile un campo per ARN del ruolo IAM. Seleziona il collegamento in questo campo per passare al ruolo IAM per questo notebook.
6. Crea la policy seguente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DeleteLogDelivery",
        "logs:Describe*",
        "logs:GetLogDelivery",
        "logs:GetLogEvents",
        "logs:ListLogDeliveries",
        "logs:PutLogEvents",
        "logs:PutResourcePolicy",
        "logs:UpdateLogDelivery"
      ],
      "Resource": "*"
    }
  ]
```

```
}
```

7. Salva questa nuova policy e collegala al ruolo IAM nel passaggio 4.
8. Seleziona Start in alto a destra nella pagina dei dettagli dell'istanza del SageMaker notebook.
9. Quando inizia il flusso dei log, visualizzerai un collegamento Visualizza log sotto il campo denominato Configurazione del ciclo di vita nella parte inferiore sinistra della sezione Impostazioni dell'istanza notebook della pagina dei dettagli.

Se il notebook non si avvia, nella pagina dei dettagli del notebook sulla SageMaker console verrà visualizzato un messaggio che indica che l'avvio dell'istanza del notebook ha impiegato più di 5 minuti. I CloudWatch registri relativi a questo problema sono disponibili sotto il nome:. *(your-notebook-name)*`/LifecycleConfigOnStart`

Se necessario, consulta Log Amazon SageMaker Events with Amazon CloudWatch per maggiori dettagli.

Utilizzo della registrazione di log delle query lente di Amazon Neptune

Identificare, eseguire il debug e ottimizzare una query a esecuzione lenta può essere difficile. Quando è abilitata la registrazione dei log delle query lente di Neptune, gli attributi di tutte le query a esecuzione prolungata vengono registrati automaticamente per semplificare questo processo.

> **Note**
>
> La registrazione di log delle query lente è stata introdotta nel rilascio 1.2.1.0 del motore Neptune.

È possibile abilitare la registrazione di log delle query lente utilizzando il parametro del cluster database neptune_enable_slow_query_log. Questo parametro è impostato su `disabled` per impostazione predefinita. L'impostazione su `info` o `debug` abilita la registrazione di log delle query lente. L'impostazione `info` registra alcuni attributi utili di ogni query a esecuzione lenta, mentre l'impostazione `debug` registra tutti gli attributi disponibili.

Per impostare la soglia per quella che è considerata una query a esecuzione lenta, utilizzare il parametro del cluster database neptune_slow_query_log_threshold per specificare il numero di

millisecondi dopo i quali una query in esecuzione viene considerata lenta e viene registrata quando è abilitata la registrazione di log delle query lente. Il valore predefinito è 5000 millisecondi (5 secondi).

Puoi impostare questi parametri del cluster DB nella o utilizzando AWS Management Console il AWS CLI comando modify-db-cluster-parameter-group o la funzione di gestione ModifyDB ClusterParameterGroup.

> **Note**
>
> I parametri di registrazione di log delle query lente sono dinamici, il che significa che la modifica dei loro valori non richiede né causa il riavvio del cluster database.

Per visualizzare i log delle interrogazioni lente in AWS Management Console

È possibile visualizzare e scaricare i log delle query lente in, come segue: AWS Management Console

Nella pagina Istanze, scegli l'istanza database, quindi scorri la pagina fino alla sezione Log. È quindi possibile selezionare un file di log e quindi scegliere Scarica per scaricarlo.

File generati dalla registrazione di log delle query lente in Neptune

I file di log generati dalla registrazione di log delle query lente in Neptune hanno le seguenti caratteristiche:

- I file sono codificati come UTF-8.
- Le query e i relativi attributi vengono registrati in formato JSON.
- Gli attributi nulli e vuoti non vengono registrati, ad eccezione dei dati `queryTime`.
- I log si estendono su più file, il cui numero varia a seconda della dimensione dell'istanza.
- Le voci dei log non sono in ordine sequenziale. Per ordinarle, puoi utilizzare il valore `timestamp`.
- Per visualizzare gli ultimi eventi, potrebbe essere necessario esaminare tutti i file di log delle query lente.
- I file di log vengono ruotati quando raggiungono la dimensione di 100 MiB in forma aggregata. Questo limite non è configurabile.

Attributi di query registrati in modalità **`info`**

I seguenti attributi vengono registrati per le query lente quando il parametro del cluster database `neptune_enable_slow_query_log` è stato impostato su `info`:

Group (Gruppo)	Attributo	Descrizione
requestResponseMetadata	`requestId`	ID della richiesta della query.
	`requestType`	Tipo di richiesta, ad esempio HTTP o. WebSocket
	`responseStatusCode`	Codice di stato della risposta alla query, ad esempio 200.
	`exceptionClass`	Classe di eccezione dell'erro re restituito dopo l'esecuzione della query.
queryStats	`query`	Stringa di query.
	`queryFingerprint`	Impronta digitale della query.
	`queryLanguage`	Linguaggio di query, come Gremlin, SPARQL o openCypher.
memoryStats	`allocatedPermits`	Autorizzazioni assegnate alla query.
	`approximateUsedMem oryBytes`	Memoria approssimativa utilizzata dalla query durante l'esecuzione.
queryTime	`startTime`	Ora di inizio della query (UTC).
	`overallRunTimeMs`	Tempo di esecuzione totale della query, in millisecondi.

Group (Gruppo)	Attributo	Descrizione
	`parsingTimeMs`	Tempo di analisi delle query, in millisecondi.
	`waitingTimeMs`	Tempo di attesa della coda della query Gremlin/SPARQL/openCypher, in millisecondi
	`executionTimeMs`	Tempo di esecuzione della query, in millisecondi.
	`serializationTimeMs`	Tempo di serializzazione delle query, in millisecondi.
statementCounters	`scanned`	Numero di istruzioni analizzate.
	`written`	Numero di istruzioni scritte.
	`deleted`	Numero di istruzioni eliminate.
transactionCounters	`committed`	Numero di transazioni sottoposte a commit.
	`rolledBack`	Numero di transazioni di cui è stato eseguito il rollback.
vertexCounters	`added`	Numero di vertici aggiunti.
	`removed`	Numero di vertici rimossi.
	`propertiesAdded`	Numero di proprietà dei vertici aggiunte.
	`propertiesRemoved`	Numero di proprietà dei vertici aggiunte.
edgeCounters	`added`	Numero di archi aggiunti.

Group (Gruppo)	Attributo	Descrizione
	`removed`	Numero di archi rimossi.
	`propertiesAdded`	Numero di proprietà degli archi aggiunte.
	`propertiesRemoved`	Numero di proprietà degli archi rimosse.
resultCache	`hitCount`	Numero di riscontri nella cache dei risultati.
	`missCount`	Numero di mancati riscontro nella cache dei risultati.
	`putCount`	Numero di inserimenti nella cache dei risultati.
concurrentExecution	`acceptedQueryCountAtStart`	Query parallele accettate con l'esecuzione della query corrente all'inizio.
	`runningQueryCountAtStart`	Query parallele in esecuzione con l'esecuzione della query corrente all'inizio.
	`acceptedQueryCountAtEnd`	Query parallele accettate con l'esecuzione della query corrente alla fine.
	`runningQueryCountAtEnd`	Query parallele in esecuzione con l'esecuzione della query corrente alla fine.
queryBatch	`queryProcessingBatchSize`	Dimensione del batch durante l'elaborazione delle query.

Group (Gruppo)	Attributo	Descrizione
	`querySerialisation BatchSize`	Dimensione del batch durante la serializzazione delle query.

Attributi di query registrati in modalità **debug**

Quando il parametro del cluster database `neptune_enable_slow_query_log` è impostato su `debug`, vengono registrati i seguenti attributi del contatore di archiviazione oltre agli attributi registrati in modalità `info`:

Attributo	Descrizione
`statementsScannedInAllIndexes`	Istruzioni analizzate in tutti gli indici.
`statementsScannedSPOGIndex`	Istruzioni analizzate nell'indice SPOG.
`statementsScannedPOGSIndex`	Istruzioni analizzate nell'indice POGS.
`statementsScannedGPSOIndex`	Istruzioni analizzate nell'indice GPSO.
`statementsScannedOSGPIndex`	Istruzioni analizzate nell'indice OSGP.
`statementsScannedInChunk`	Istruzioni analizzate insieme in blocchi.
`postFilteredStatementScans`	Istruzioni rimaste dopo il post-filtraggio dopo le analisi.
`distinctStatementScans`	Istruzioni distinte analizzate.
`statementsReadInAllIndexes`	Istruzioni lette dopo l'analisi post-filtraggio in tutti gli indici.
`statementsReadSPOGIndex`	Istruzioni lette dopo l'analisi post-filtraggio nell'indice SPOG.
`statementsReadPOGSIndex`	Istruzioni lette dopo l'analisi post-filtraggio nell'indice POGS.

Attributo	Descrizione
`statementsReadGPSOIndex`	Istruzioni lette dopo l'analisi post-filtraggio nell'indice GPSO.
`statementsReadOSGPIndex`	Istruzioni lette dopo l'analisi post-filtraggio nell'indice OSGP.
`accessPathSearches`	Numero di ricerche del percorso di accesso.
`fullyBoundedAccessPathSearches`	Numero di ricerche del percorso di accesso con chiavi completamente limitate.
`accessPathSearchedByPrefix`	Numero di ricerche del percorso di accesso per prefisso.
`searchesWhereRecordsWereFound`	Numero di ricerche con 1 o più record come output.
`searchesWhereRecordsWereNotFound`	Numero di ricerche senza record come output.
`totalRecordsFoundInSearches`	Record totali trovati da tutte le ricerche.
`statementsInsertedInAllIndexes`	Numero di istruzioni inserite in tutti gli indici.
`statementsUpdatedInAllIndexes`	Numero di istruzioni aggiornate in tutti gli indici.
`statementsDeletedInAllIndexes`	Numero di istruzioni eliminate in tutti gli indici.
`predicateCount`	Numero di predicati.
`dictionaryReadsFromValueToIdTable`	Numero di letture del dizionario dal valore della tabella ID.
`dictionaryReadsFromIdToValueTable`	Numero di letture del dizionario dall'ID della tabella valori.
`dictionaryWritesToValueToIdTable`	Numero di scritture del dizionario nel valore della tabella ID.

Attributo	Descrizione
`dictionaryWritesToIdToValueTable`	Numero di scritture del dizionario nell'ID della tabella valori.
`rangeCountsInAllIndexes`	Numero di conteggio intervalli in tutti gli indici.
`deadlockCount`	Numero di deadlock nella query.
`singleCardinalityInserts`	Numero di inserimenti a cardinalità singola eseguiti.
`singleCardinalityInsertDeletions`	Numero di istruzioni eliminate durante un inserimento a cardinalità singola.

Esempio di registrazione di log di debug per una query lenta

L'esecuzione della seguente query Gremlin potrebbe richiedere più tempo rispetto alla soglia impostata per le query lente:

```
gremlin=g.V().has('code','AUS').repeat(out().simplePath()).until(has('code','AGR')).path().by('
```

Quindi, se si abilita la registrazione di log delle query lente in modalità debug, verranno registrati i seguenti attributi per la query, in un formato simile a quello riportato di seguito:

```
{
  "requestResponseMetadata": {
    "requestId": "5311e493-0e98-457e-9131-d250a2ce1e12",
    "requestType": "HTTP_GET",
    "responseStatusCode": 200
  },
  "queryStats": {
    "query":
 "gremlin=g.V().has('code','AUS').repeat(out().simplePath()).until(has('code','AGR')).path().by
    "queryFingerprint":
 "g.V().has(string0,string1).repeat(__.out().simplePath()).until(__.has(string0,string2)).path(
    "queryLanguage": "Gremlin"
  },
  "memoryStats": {
    "allocatedPermits": 20,
```

```
    "approximateUsedMemoryBytes": 14838
  },
  "queryTimeStats": {
    "startTime": "23/02/2023 11:42:52.657",
    "overallRunTimeMs": 2249,
    "executionTimeMs": 2229,
    "serializationTimeMs": 13
  },
  "statementCounters": {
    "read": 69979
  },
  "transactionCounters": {
    "committed": 1
  },
  "concurrentExecutionStats": {
    "acceptedQueryCountAtStart": 1
  },
  "queryBatchStats": {
    "queryProcessingBatchSize": 1000,
    "querySerialisationBatchSize": 1000
  },
  "storageCounters": {
    "statementsScannedInAllIndexes": 69979,
    "statementsScannedSPOGIndex": 44936,
    "statementsScannedPOGSIndex": 4,
    "statementsScannedGPSOIndex": 25039,
    "statementsReadInAllIndexes": 68566,
    "statementsReadSPOGIndex": 43544,
    "statementsReadPOGSIndex": 2,
    "statementsReadGPSOIndex": 25020,
    "accessPathSearches": 27,
    "fullyBoundedAccessPathSearches": 27,
    "dictionaryReadsFromValueToIdTable": 10,
    "dictionaryReadsFromIdToValueTable": 17,
    "rangeCountsInAllIndexes": 4
  }
}
```

Registrazione delle chiamate API Amazon Neptune con AWS CloudTrail

Amazon Neptune è integrato AWS CloudTrail con, un servizio che fornisce un registro delle azioni intraprese da un utente, ruolo o AWS servizio in Neptune. CloudTrail acquisisce le chiamate API per Neptune come eventi, incluse le chiamate dalla console Neptune e le chiamate in codice alle API Neptune.

CloudTrail registra solo gli eventi per le chiamate API di gestione di Neptune, come la creazione di un'istanza o di un cluster. Per controllare le modifiche al tuo grafo, puoi utilizzare i log di audit. Per ulteriori informazioni, consulta Utilizzo dei log di audit con i cluster Amazon Neptune.

> **Important**
>
> La console AWS CLI di Amazon Neptune e le chiamate API vengono registrate come chiamate effettuate all'API Amazon Relational Database Service (Amazon RDS).

Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Neptune. Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare la richiesta che è stata effettuata a Neptune, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e dettagli aggiuntivi.

Per ulteriori informazioni CloudTrail, consulta la Guida per l'AWS CloudTrail utente.

Informazioni su Nettuno in CloudTrail

CloudTrail è abilitato sul tuo AWS account al momento della creazione dell'account. Quando si verifica un'attività in Amazon Neptune, tale attività viene registrata in CloudTrail un evento insieme ad AWS altri eventi di servizio nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare eventi recenti nel tuo AWS account. Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi nel tuo AWS account, inclusi gli eventi per Neptune, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per

impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le Regioni . Il trail registra gli eventi di tutte le regioni della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consultare:

- Panoramica della creazione di un trail
- CloudTrail Servizi e integrazioni supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Se viene eseguita un'azione per conto del tuo AWS account utilizzando la console Neptune, l'interfaccia a riga di comando di Neptune o le API Neptune SDK, registra l'azione come chiamate effettuate all'API Amazon RDS AWS CloudTrail . Ad esempio, se utilizzi la console Neptune per modificare un'istanza DB o richiamare AWS CLImodify-db-instanceil comando, AWS CloudTrail il log mostra una chiamata all'azione ModifyDBInstance dell'API Amazon RDS. Per un elenco delle azioni dell'API Neptune registrate da AWS CloudTrail, consulta il Neptune API Reference.

> **Note**
>
> AWS CloudTrail registra solo gli eventi per le chiamate API di gestione di Neptune, come la creazione di un'istanza o di un cluster. Per controllare le modifiche al tuo grafo, puoi utilizzare i log di audit. Per ulteriori informazioni, consulta Utilizzo dei log di audit con i cluster Amazon Neptune.

Ogni evento o voce di registro contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali dell'utente IAM o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro servizio. AWS

Per ulteriori informazioni, vedete l'elemento CloudTrail userIdentity.

Informazioni sulle voci dei file di log Neptune

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra un CloudTrail registro per un utente che ha creato uno snapshot di un'istanza DB e poi l'ha eliminata utilizzando la console Neptune. La console è identificata dall'elemento `userAgent`. Le chiamate API richieste effettuate dalla console (`CreateDBSnapshot` e `DeleteDBInstance`) si trovano nell'elemento `eventName` di ciascun record. Le informazioni relative all'utente (`Alice`) sono disponibili nell'elemento `userIdentity`.

```
{
  Records:[
  {
    "awsRegion":"us-west-2",
    "eventName":"CreateDBSnapshot",
    "eventSource":"",
    "eventTime":"2014-01-14T16:23:49Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-
kaos_fleet-1108-prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
```

```
    }
  },
  {
    "awsRegion":"us-west-2",
    "eventName":"DeleteDBInstance",
    "eventSource":"",
    "eventTime":"2014-01-14T16:28:27Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-
kaos_fleet-1108-prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
    }
  }
  ]
}
```

Utilizzo della notifica di eventi Neptune

Argomenti

- Categorie di eventi Amazon Neptune e messaggi di evento
- Sottoscrizione alle notifiche eventi Neptune
- Gestione delle sottoscrizioni alle notifiche eventi Neptune

Amazon Neptune usa Amazon Simple Notification Service (Amazon SNS) per fornire le notifiche quando si verifica un evento Neptune. Queste notifiche possono essere in qualsiasi forma supportata da Amazon SNS per una AWS regione, ad esempio un'e-mail, un messaggio di testo o una chiamata a un endpoint HTTP.

Neptune raggruppa questi eventi in categorie che puoi sottoscrivere per ricevere una notifica quando si verifica un evento di tale categoria. Puoi sottoscrivere una categoria di eventi per un'istanza database, un cluster database, uno snapshot DB, uno snapshot cluster database o un gruppo di parametri database. Ad esempio, sottoscrivendo la categoria Backup per una determinata istanza database, riceverai una notifica ogni volta che si verifica un evento relativo al backup che interessa l'istanza database. Riceverai una notifica anche quando viene modificata la sottoscrizione a una notifica eventi.

Gli eventi si verificano sia a livello di cluster database che di istanza database, quindi puoi ricevere gli eventi se esegui la sottoscrizione a un cluster database o a un'istanza database.

Le notifiche eventi vengono inviate all'indirizzo fornito al momento della creazione della sottoscrizione. È possibile creare più sottoscrizioni diverse, ad esempio una che riceve notifiche per tutti gli eventi e un'altra che include solo gli eventi critici per le istanze database di produzione. Puoi disattivare facilmente la notifica senza eliminare una sottoscrizione. A tale scopo, imposta il pulsante di opzione Abilitato su No nella console Neptune.

> **Important**
>
> Amazon Neptune non garantisce l'ordine degli eventi inviati in un flusso di eventi. Tale ordine è soggetto a modifiche.

Neptune usa il nome della risorsa Amazon (ARN) di un argomento Amazon SNS per identificare ogni sottoscrizione. La console Neptune crea automaticamente l'ARN quando crei la sottoscrizione.

La fatturazione per la notifica degli eventi Neptune avviene tramite Amazon SNS. L'uso della notifica degli eventi è soggetta alle tariffe di Amazon SNS. Per ulteriori informazioni, consulta Prezzi di Amazon Simple Notification Service.

Categorie di eventi Amazon Neptune e messaggi di evento

Neptune genera un numero significativo di eventi in categorie a cui puoi effettuare la sottoscrizione tramite la console Neptune. Ogni categoria si applica a un tipo di origine, ad esempio un'istanza database, uno snapshot DB o un gruppo di parametri database.

> **Note**
>
> Neptune utilizza le definizioni e gli ID degli eventi Amazon RDS esistenti.

Eventi Neptune provenienti da istanze database

La tabella seguente riporta un elenco di eventi per categoria di eventi applicabili quando il tipo di origine è un'istanza database.

Categoria	ID evento Amazon RDS	Descrizione
availability	RDS-EVENT-0006	L'istanza database è stata riavviata.
	RDS-EVENT-0004	L'istanza database è stata arrestata.
	RDS-EVENT-0022	Si è verificato un errore durante il riavvio del motore Neptune.
backup	RDS-EVENT-0001	Viene eseguito il backup dell'istanza database.
	RDS-EVENT-0002	Il backup dell'istanza database è terminato.
modifica della configurazione	RDS-EVENT-0009	L'istanza database è stata aggiunta a un gruppo di sicurezza.
	RDS-EVENT-0024	È in corso la conversione dell'istanza database in un'istanza database Multi-AZ.

Categoria	ID evento Amazon RDS	Descrizione
	RDS-EVENT-0030	È in corso la conversione dell'istanza database in un'istanza database Single-AZ.
	RDS-EVENT-0012	Vengono applicate le modifiche alla classe di istanza database.
	RDS-EVENT-0018	È in corso la modifica delle impostazioni di storage correnti per l'istanza database.
	RDS-EVENT-0011	Un gruppo di parametri dell'istanza database è stato modificato.
	RDS-EVENT-0092	È stato ultimato l'aggiornamento di un gruppo di parametri dell'istanza database.
	RDS-EVENT-0028	I backup automatici per l'istanza database sono stati disabilitati.
	RDS-EVENT-0032	I backup automatici per l'istanza database sono stati abilitati.

Categoria	ID evento Amazon RDS	Descrizione
	RDS-EVENT-0025	L'istanza database è stata convertita in un'istanza database Multi-AZ.
	RDS-EVENT-0029	L'istanza database è stata convertita in un'istanza database Single-AZ.
	RDS-EVENT-0014	La classe di istanza database per l'istanza database è stata modificata.
	RDS-EVENT-0017	Le impostazioni di storage correnti per l'istanza database sono state modificate.
	RDS-EVENT-0010	L'istanza database è stata rimossa da un gruppo di sicurezza.
creazione	RDS-EVENT-0005	Viene creata l'istanza database.
eliminazione	RDS-EVENT-0003	L'istanza database è stata eliminata.
failover	RDS-EVENT-0034	Neptune non sta tentando di effettuare un failover richiesto perché di recente si è verificato un failover nell'istanza database.

Categoria	ID evento Amazon RDS	Descrizione
	RDS-EVENT-0013	È stato avviato un failover Multi-AZ che ha portato alla promozione di un'istanza in stand-by.
	RDS-EVENT-0015	È stato completato un failover Multi-AZ che ha portato alla promozione di un'istanza in stand-by. Potrebbero essere necessari alcuni minuti per il trasferimento del DNS alla nuova istanza database principale.
	RDS-EVENT-0065	L'istanza è stata recuperata da un failover parziale.
	RDS-EVENT-0049	È stato completato un failover Multi-AZ.
	RDS-EVENT-0050	È stata avviata un'attivazione Multi-AZ dopo il recupero dell'istanza.
	RDS-EVENT-0051	È stata completata un'attivazione Multi-AZ. Ora il database dovrebbe essere accessibile.

Categoria	ID evento Amazon RDS	Descrizione
	RDS-EVENT-0031	L'istanza database ha avuto esito negativo a causa di una configurazione non compatibile o di un problema di storage sottostante. Inizia a point-in-time-restore per l'istanza DB.
	RDS-EVENT-0036	L'istanza database si trova in una rete non compatibile. Alcuni degli ID sottorete specificati non sono validi o non esistono.
	RDS-EVENT-0035	L'istanza database include parametri non validi. Ad esempio, se non è stato possibile avviare l'istanza database perché un parametro relativo alla memoria è troppo elevato per questa classe di istanze, il cliente dovrebbe modificare il parametro di memoria e riavviare l'istanza database.

Categoria	ID evento Amazon RDS	Descrizione
	RDS-EVENT-0082	Neptune non è riuscito a copiare i dati di backup da un bucket Amazon S3. È probabile che le autorizzazioni necessarie perché Neptune possa accedere al bucket Amazon S3 non siano configurate correttamente.
storage insufficiente	RDS-EVENT-0089	L'istanza database ha utilizzato oltre il 90% dello storage allocato. Puoi monitorare lo spazio di storage per un'istanza database usando il parametro Free Storage Space (Spazio di storage libero).
	RDS-EVENT-0007	Lo storage allocato per l'istanza database è esaurito. Per risolvere questo problema, è necessario allocare ulteriore spazio di storage per l'istanza database.

Categoria	ID evento Amazon RDS	Descrizione
manutenzione	RDS-EVENT-0026	È in corso la manutenzione offline dell'istanza database. L'istanza database non è attualmente disponibile.
	RDS-EVENT-0027	La manutenzione offline dell'istanza database è stata completata. L'istanza database è ora disponibile.
	RDS-EVENT-0047	L'applicazione di patch all'istanza database è stata completata.
notifica	RDS-EVENT-0044	Notifica emessa dall'operatore. Per ulteriori informazioni, consulta il messaggio di evento.
	RDS-EVENT-0048	L'applicazione di patch all'istanza database è stata ritardata.
	RDS-EVENT-0087	L'istanza database è stata arrestata.
	RDS-EVENT-0088	L'istanza database è stata avviata.

Categoria	ID evento Amazon RDS	Descrizione
	RDS-EVENT-0154	L'istanza database è in fase di avvio poiché supera il tempo massimo concesso per l'arresto.
	RDS-EVENT-0158	L'istanza database è in uno stato che non può essere aggiornato.
	RDS-EVENT-0173	È stata applicata una patch all'istanza database.
replica di lettura	RDS-EVENT-0045	Si è verificato un errore interno nel processo di replica di lettura. Per ulteriori informazioni, consulta il messaggio di evento.

Categoria	ID evento Amazon RDS	Descrizione
	RDS-EVENT-0046	La replica di lettura ha ripristinato la replica. Questo messaggio viene visualizzato quando crei per la prima volta una replica di lettura o come messaggio di monitoraggio che conferma il corretto funzionamento della replica. Se il messaggio segue una notifica RDS-EVENT -0045, la replica è ripresa a seguito di un errore o dopo l'arresto della replica.
	RDS-EVENT-0057	La replica è stata terminata nella replica di lettura.
	RDS-EVENT-0062	La replica è stata arrestata manualmen te nella replica di lettura.
	RDS-EVENT-0063	La replica è stata reimpostata nella replica di lettura.

Categoria	ID evento Amazon RDS	Descrizione
recupero	RDS-EVENT-0020	È stato avviato il recupero dell'istanza database. La durata del recupero varia in funzione della quantità di dati da recuperare.
	RDS-EVENT-0021	È stato completato il recupero dell'istanza database.
	RDS-EVENT-0023	È stato richiesto un backup manuale, ma in Neptune è in corso la creazione di uno snapshot database. Inviare di nuovo la richiesta quando Neptune avrà completato lo snapshot database.
	RDS-EVENT-0052	È stato avviato il recupero dell'istanza Multi-AZ. La durata del recupero varia in funzione della quantità di dati da recuperare.
	RDS-EVENT-0053	È stato completato il recupero dell'istanza Multi-AZ.

Categoria	ID evento Amazon RDS	Descrizione
ripristino	RDS-EVENT-0008	L'istanza database è stata ripristinata da uno snapshot DB.
	RDS-EVENT-0019	L'istanza DB è stata ripristinata da un point-in-time backup.

Eventi Neptune provenienti da un cluster database

La tabella seguente riporta un elenco di eventi per categoria di eventi applicabili quando il tipo di origine è un cluster database.

Categoria	ID evento RDS	Descrizione
failover	RDS-EVENT-0069	Il failover del cluster di database non è riuscito.
	RDS-EVENT-0070	Il failover del cluster di database è stato riavviato.
	RDS-EVENT-0071	Il failover del cluster di database è stato ultimato.
	RDS-EVENT-0072	Il failover del cluster di database è iniziato nella stessa zona di disponibilità.
	RDS-EVENT-0073	Il failover del cluster di database è iniziato in

Categoria	ID evento RDS	Descrizione
		più zone di disponibilità.
	RDS-EVENT-0083	Neptune non è riuscito a copiare i dati di backup da un bucket Amazon S3. È probabile che le autorizzazioni necessarie perché Neptune possa accedere al bucket Amazon S3 non siano configurate correttamente.
manutenzione	RDS-EVENT-0156	È disponibile un aggiornamento della versione secondaria del motore del database per il cluster di database.
notifica	RDS-EVENT-0076	La migrazione a un cluster database Neptune non è riuscita.

Categoria	ID evento RDS	Descrizione
	RDS-EVENT-0077	Un tentativo di convertire una tabella dal database di origine al modulo database non è riuscito durante la migrazione a un cluster database Neptune.
	RDS-EVENT-0150	Il cluster di database è stato arrestato.
	RDS-EVENT-0151	Il cluster di database è stato avviato.
	RDS-EVENT-0152	Impossibile interrompere il cluster di database.
	RDS-EVENT-0153	Il cluster di database è in fase di avvio poiché supera il tempo massimo concesso per l'arresto.

Eventi Neptune provenienti da uno snapshot del cluster database

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è uno snapshot del cluster database Neptune.

Categoria	ID evento RDS	Descrizione
backup	RDS-EVENT-0074	È stata avvia la creazione di uno snapshot cluster di database manuale.
backup	RDS-EVENT-0075	È stato creato uno snapshot cluster di database manuale.
notifica	RDS-EVENT-0162	Attività di esportazione snapshot cluster di database non riuscita.
notifica	RDS-EVENT-0163	Attività di esportazione snapshot cluster di database annullata.
notifica	RDS-EVENT-0164	Attività di esportazione snapshot cluster di database completata.
backup	RDS-EVENT-0168	Creazione snapshot cluster automatizzato.
backup	RDS-EVENT-0169	Snapshot di cluster automatizzato creato.
creazione	RDS-EVENT-0170	Cluster di database creato.
eliminazione	RDS-EVENT-0171	Cluster di database eliminato.
notifica	RDS-EVENT-0172	Rinominato cluster di database da [vecchio

Categoria	ID evento RDS	Descrizione
		nome cluster di database] a [nuovo nome cluster di database].

Eventi Neptune provenienti da un gruppo di parametri del cluster database

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un gruppo di parametri del cluster database.

Categoria	ID evento RDS	Descrizione
modifica della configurazione	RDS-EVENT-0037	Il gruppo di parametri è stato modificato.

Eventi Neptune provenienti da un gruppo di sicurezza

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un gruppo di sicurezza.

Categoria	ID evento RDS	Descrizione
modifica della configurazione	RDS-EVENT-0038	Il gruppo di sicurezza è stato modificato.
errore	RDS-EVENT-0039	Il gruppo di sicurezza appartenente a [utente] non esiste. L'autorizzazione per il gruppo di sicurezza è stata revocata.

Sottoscrizione alle notifiche eventi Neptune

Puoi usare la console Neptune per sottoscrivere le notifiche degli eventi, come segue:

Per sottoscrivere una notifica eventi Neptune

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel pannello di navigazione selezionare Event subscriptions (Sottoscrizioni di eventi).
3. Nel riquadro Event subscriptions (Sottoscrizioni di eventi) scegliere Create event subscription (Crea sottoscrizione di eventi).
4. Nella finestra di dialogo Create event subscription (Crea sottoscrizione di eventi), seguire questa procedura:
    a. Per Name (Nome), immettere un nome per la sottoscrizione alle notifiche eventi.
    b. Per Send notifications to (Invia notifiche a), scegliere un ARN Amazon SNS esistente per un argomento Amazon SNS oppure scegliere create topic (crea argomento) per immettere il nome di un argomento e un elenco di destinatari.
    c. Per Source type (Tipo di origine) scegliere un tipo di origine.
    d. Scegliere Yes (Sì) per abilitare la sottoscrizione. Per creare una sottoscrizione senza ricevere subito le notifiche, scegliere No.
    e. A seconda del tipo di origine selezionato, scegliere le categorie di eventi e le origini da cui ricevere le notifiche eventi.
    f. Scegli Crea.

Gestione delle sottoscrizioni alle notifiche eventi Neptune

Se scegli Sottoscrizioni a eventi nel riquadro di navigazione della console Neptune, puoi visualizzare le categorie di sottoscrizione e un elenco delle sottoscrizioni correnti.

Puoi anche modificare o eliminare una sottoscrizione specifica.

Modifica delle sottoscrizioni alle notifiche eventi Neptune

Per modificare una sottoscrizione alle notifiche eventi Neptune corrente

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel pannello di navigazione selezionare Event subscriptions (Sottoscrizioni di eventi). Il riquadro Event subscriptions (Sottoscrizioni di eventi) mostra tutte le sottoscrizioni delle notifiche degli eventi.
3. Nel riquadro Event subscriptions (Sottoscrizioni di eventi) scegliere la sottoscrizione da modificare e selezionare Edit (Modifica).
4. Apportare le modifiche alla sottoscrizione nella sezione Target (Destinazione) o Source (Origine). Puoi aggiungere o rimuovere gli identificatori di origine selezionandoli o deselezionandoli nella sezione Origine.
5. Scegli Modifica. La console Neptune indica che è in corso la modifica della sottoscrizione.

Eliminazione di una sottoscrizione alle notifiche eventi Neptune

Puoi eliminare un abbonamento quando questo non è più necessario. Tutti gli abbonati all'argomento non riceveranno più le notifiche di eventi specificate dall'abbonamento.

Per eliminare una sottoscrizione alle notifiche eventi Neptune

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel pannello di navigazione selezionare Event subscriptions (Sottoscrizioni di eventi).
3. Nel riquadro Sottoscrizione a eventi scegliere la sottoscrizione che si vuole eliminare.
4. Scegli Elimina.
5. La console Neptune indica che è in corso l'eliminazione della sottoscrizione.

Applicazione di tag alle risorse Amazon Neptune

È possibile usare i tag di Neptune per aggiungere metadati alle risorse di Neptune. Inoltre, puoi utilizzare tag con policy AWS Identity and Access Management (IAM) per gestire l'accesso alle risorse di Neptune e controllare quali azioni possono essere applicate a tali risorse. Infine, i tag possono essere utilizzati per monitorare i costi raggruppando le spese per risorse con tag simili.

È possibile applicare tag a tutte le risorse amministrative di Neptune, tra cui:

- Istanze DB
- Cluster database
- Repliche di lettura
- Snapshot DB
- Snapshot cluster database
- Abbonamenti a eventi
- Gruppi di parametri database
- Gruppi di parametri di cluster database
- Gruppi di sottoreti database

Panoramica sui tag delle risorse di Neptune

Un tag di Amazon Neptune è una coppia nome-valore definita e associata alla risorsa di Neptune. Il nome viene definito chiave. L'indicazione di un valore per la chiave è un'operazione facoltativa. È possibile usare i tag per assegnare informazioni arbitrarie a una risorsa di Neptune. Una chiave tag potrebbe essere impiegata, ad esempio, per definire una categoria e il valore di tag potrebbe essere un elemento di tale categoria. Ad esempio, puoi definire una chiave di tag "progetto" e un valore di tag "Salix", che indica che la risorsa Neptune viene assegnata al progetto Salix. È anche possibile usare i tag per indicare le risorse di Neptune usate a scopo di test o produzione tramite una chiave, ad esempio `environment=test` o `environment=production`. È consigliabile utilizzare un set coerente di chiavi di tag per agevolare il monitoraggio dei metadati associati alle risorse di Neptune.

Utilizzate i tag per organizzare la AWS fattura in modo da rispecchiare la vostra struttura dei costi. A tale scopo, registrati per ricevere la Account AWS fattura con i valori chiave dell'etichetta inclusi. Per visualizzare il costo delle risorse combinate, puoi organizzare le informazioni di fatturazione in base alle risorse con gli stessi valori di chiave di tag. Puoi ad esempio applicare tag a numerose risorse con un nome di applicazione specifico, quindi organizzare le informazioni di fatturazione per visualizzare il costo totale dell'applicazione in più servizi. Per ulteriori informazioni, consulta la pagina sull'utilizzo dei tag per l'allocazione dei costi nella Guida per l'utente di AWS Billing .

Ogni risorsa di Neptune dispone di un set di tag contenente tutti i tag assegnati a tale risorsa di Neptune. Un set di tag può contenere fino a 10 tag ma può anche essere vuoto. Se aggiungi un

tag a una risorsa di Neptune con la stessa chiave di un tag esistente sulla risorsa, il nuovo valore sovrascrive quello precedente.

AWS non applica alcun significato semantico ai tag; i tag vengono interpretati rigorosamente come stringhe di caratteri. Neptune può impostare i tag in un'istanza database o in altre risorse di Neptune, a seconda delle impostazioni scelte al momento della creazione della risorsa. Ad esempio, Neptune potrebbe aggiungere un tag che indica che un'istanza database viene utilizzata solo a scopo di test o produzione.

- La chiave di tag corrisponde al nome obbligatorio del tag. Il valore della stringa può essere composto da 1 a 128 caratteri Unicode e non può avere il prefisso "`aws:`" o "`rds:`". La stringa può contenere solo il set di lettere, cifre, spazi vuoti, "_", ".", "/", "=", "+", "-" Unicode (espressioni regolari Java: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`").
- Il valore di tag è un valore di stringa opzionale del tag. Il valore della stringa può essere composto da 1 a 256 caratteri Unicode e non può avere il prefisso "`aws:`". La stringa può contenere solo il set di lettere, cifre, spazi vuoti, "_", ".", "/", "=", "+", "-" Unicode (espressioni regolari Java: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`").

  I valori non devono essere necessariamente univoci in un set di tag e possono essere Null. Ad esempio, puoi avere una coppia chiave-valore in un set di tag `project/Trinity` e `cost-center/Trinity`.

> **Note**
>
> È possibile aggiungere un tag a una snapshot. Tuttavia, l'addebito non rifletterà questo raggruppamento.

È possibile utilizzare l' AWS Management Console API Neptune o la Neptune per aggiungere, elencare ed eliminare tag sulle risorse Neptune. AWS CLI Quando utilizzi l'API AWS CLI o l'API Neptune, devi fornire l'Amazon Resource Name (ARN) per la risorsa Neptune con cui desideri lavorare. Per ulteriori informazioni sulla creazione di un ARN, consultare Costruzione di un ARN per Neptune.

I tag sono memorizzati nella cache a fini di autorizzazione. Questo è il motivo per il quale le aggiunte e gli aggiornamenti dei tag delle risorse di Neptune potrebbero richiedere diversi minuti prima di diventare disponibili.

Copia dei tag in Neptune

Quando si crea o si ripristina un'istanza database, è possibile specificare che i tag dell'istanza database vengano copiati nelle snapshot dell'istanza database. La copia dei tag garantisce che i metadati delle snapshot DB corrispondano a quelli dell'istanza database di origine e che anche le policy di accesso alla snapshot DB corrispondano a quelle dell'istanza database di origine. I tag non vengono copiati per impostazione predefinita.

È possibile specificare che i tag vengano copiati nelle snapshot DB per le seguenti azioni:

- Creazione di un'istanza database.
- Ripristino di un'istanza database.
- Creazione di una replica di lettura.
- Copia di una snapshot DB.

> **Note**
>
> Se includi un valore per il `--tag-key` parametro del create-db-cluster-snapshot AWS CLI comando (o fornisci almeno un tag all'azione CreateDBClusterSnapshot API), Neptune non copia i tag dall'istanza DB di origine alla nuova istantanea del database. Questo comportamento si applica anche se l'istanza database di origine ha l'opzione `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`) abilitata.
> In questo caso puoi creare una copia di un'istanza database da una snapshot DB ed evitare l'aggiunta di tag che non si applicano alla nuova istanza database. Dopo aver creato lo snapshot DB utilizzando il AWS CLI `create-db-cluster-snapshot` comando (o l'`CreateDBClusterSnapshot`azione dell'API Neptune), è possibile aggiungere tag come descritto più avanti in questo argomento.

Etichettare Neptune usando il AWS Management Console

Il processo di applicazione dei tag a una risorsa di Amazon Neptune è simile per tutte le risorse. Di seguito viene mostrato come applicare i tag a un'istanza database Neptune.

Aggiunta di un tag a un'istanza database

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.

2. Nel riquadro di navigazione, seleziona Istanze.

> **Note**
>
> Per filtrare l'elenco di istanze database nel riquadro Instances (Istanze), digitare una stringa di testo nella casella Filter instances (Filtra istanze). Vengono visualizzate solo le istanze database che contengono la stringa.

3. Scegli l'istanza database a cui applicare i tag.
4. Scegli Instance actions (Operazioni istanza) e successivamente See details (Visualizza dettagli).
5. Nella sezione dei dettagli, scorrere verso il basso fino alla sezione Tags (Tag).
6. Scegliere Aggiungi. Viene visualizzata la finestra Add tags (Aggiungi tag).
7. Digita un valore per Tag key (Chiave tag) e Value (Valore).
8. Per aggiungere un altro tag, scegliere Add another Tag (Aggiungi un altro tag) e digitare un valore per Tag key (Chiave tag) e Value (Valore).

   Ripetere questa operazione tutte le volte necessarie.
9. Scegliere Aggiungi.

Eliminazione di un tag da un'istanza database

1. Accedi alla console di AWS gestione e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, seleziona Istanze.

> **Note**
>
> Per filtrare l'elenco di istanze database nel riquadro Instances (Istanze), digitare una stringa di testo nella casella Filter instances (Filtra istanze). Vengono visualizzate solo le istanze database che contengono la stringa.

3. Scegli l'istanza database a cui applicare i tag.
4. Scegli Instance actions (Operazioni istanza) e successivamente See details (Visualizza dettagli).
5. Nella sezione dei dettagli, scorrere verso il basso fino alla sezione Tags (Tag).
6. Scegli il tag da eliminare.

7. Scegli Remove (Rimuovi) e successivamente Remove (Rimuovi) nella finestra Remove tags (Rimuovi tag).

Etichettare Neptune usando il AWS CLI

È possibile aggiungere, elencare o rimuovere i tag per un'istanza database in Neptune utilizzando l' AWS CLI.

- Per aggiungere uno o più tag a una risorsa Neptune, usa il comando. AWS CLI `add-tags-to-resource`
- Per elencare i tag su una risorsa Neptune, usa il comando. AWS CLI `list-tags-for-resource`
- Per rimuovere uno o più tag da una risorsa Neptune, usa il comando. AWS CLI `remove-tags-from-resource`

Per ulteriori informazioni su come creare la richiesta Amazon Resource Name (ARN), consultare Costruzione di un ARN per Neptune.

Applicazione di tag in Neptune usando l'API

È possibile aggiungere, elencare o rimuovere i tag per un'istanza database utilizzando l'API di Neptune.

- Per aggiungere un tag a una risorsa di Neptune, utilizza l'operazione `AddTagsToResource`.
- Per elencare i tag assegnati a una risorsa di Neptune, utilizza l'operazione `ListTagsForResource`.
- Per rimuovere i tag da una risorsa di Neptune, utilizza l'operazione `RemoveTagsFromResource`.

Per ulteriori informazioni su come creare l'ARN necessario, consultare Costruzione di un ARN per Neptune.

Quando utilizzi XML con l'API di Neptune i tag seguono questo schema:

```
<Tagging>
    <TagSet>
        <Tag>
            <Key>Project</Key>
            <Value>Trinity</Value>
        </Tag>
```

```
        <Tag>
            <Key>User</Key>
            <Value>Jones</Value>
        </Tag>
    </TagSet>
</Tagging>
```

La tabella riportata di seguito fornisce un elenco dei tag XML consentiti e le relative caratteristiche. I valori relativi a `Key` e `Value` fanno distinzione tra maiuscole e minuscole. Ad esempio, `project=Trinity` e `PROJECT=Trinity` sono due tag distinti.

Elemento del tagging	Descrizione
`TagSet`	Un set di tag è un container di tutti i tag assegnati a una risorsa Neptune. Ogni risorsa può disporre di un solo set di tag. Puoi utilizzare `TagSet` solo con l'API di Neptune.
`Tag`	Un tag è una coppia chiave-valore definita dall'utente. Un set di tag può contenere da 1 a 50 tag.
`Key`	Una chiave corrisponde al nome obbligatorio del tag. Il valore della stringa può essere composto da 1 a 128 caratteri Unicode e non può avere il prefisso "`rds:`" o "`aws:`". La stringa può contenere solo il set di lettere, cifre, spazi vuoti, "_", ".", "/", "=", "+", "-" Unicode (espressioni regolari Java: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`"). Le chiavi devono essere uniche per un set di tag. Ad esempio, non puoi avere una coppia di chiavi in un set di tag con la stessa chiave, ma con valori diversi, ad esempio `project/Trinity` e `project/Xanadu`.
Valore	Un valore è il valore opzione del tag. Il valore della stringa può essere composto da 1 a 256 caratteri Unicode e non può avere il prefisso "`rds:`" o "`aws:`". La stringa può contenere solo il set di lettere, cifre, spazi vuoti, "_", ".", "/", "=", "+", "-" Unicode (espressioni regolari Java: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`"). I valori non devono essere necessariamente univoci in un set di tag e possono essere Null. Ad esempio, puoi avere una coppia chiave-valore in un set di tag `project/Trinity` e `cost-center/Trinity`.

Utilizzo degli ARN amministrativi in Amazon Neptune

Le risorse create in Amazon Web Services sono identificate in modo univoco con un nome della risorsa Amazon (ARN). Per determinate operazioni Amazon Neptune è necessario identificare in modo univoco una risorsa Neptune specificandone l'ARN.

> **Important**
>
> Amazon Neptune condivide il formato degli ARN di Amazon RDS per le azioni amministrative che utilizzano la Documentazione di riferimento delle API di gestione. Gli ARN amministrativi di Neptune contengono `rds` e non contengono `neptune-db`. Per gli ARN del piano dati che identificano le risorse dati Neptune, consulta Specificare le risorse dati.

Argomenti

Costruzione di un ARN per Neptune

È possibile creare un ARN per una risorsa Amazon Neptune utilizzando la sintassi seguente. Notare che Neptune condivide il formato degli ARN di Amazon RDS.

`arn:aws:rds:<region>:<account number>:<resourcetype>:<name>`

La tabella riportata di seguito mostra il formato da utilizzare quando si crea un ARN per un determinato tipo di risorsa amministrativa di Neptune.

Tipo di risorsa	Formato ARN
Istanza database	arn:aws:rds:*<region>*:*<account>* :db:*<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :db:my-instance-1`
Cluster DB	arn:aws:rds:*<region>*:*<account>* :cluster: *<name>*

Tipo di risorsa	Formato ARN
	Per esempio: `arn:aws:rds: us-east-2 :123456789012 :cluster: my-cluster-1`
Sottoscrizione a eventi	arn:aws:rds:*<region>*:*<account>* :es:*<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :es:my-subscription`
DB parameter group (Gruppo di parametri database)	arn:aws:rds:*<region>*:*<account>* :pg:*<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :pg:my-param-enable-logs`
Gruppo di parametri del cluster DB	arn:aws:rds:*<region>*:*<account>* :cluster-pg: *<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :cluster-pg: my-cluster-param-timezone`
Snapshot cluster di database	arn:aws:rds:*<region>*:*<account>* :cluster-snapshot: *<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-snap-20160809`

Tipo di risorsa	Formato ARN
Gruppo di sottoreti DB	arn:aws:rds:*<region>*:*<account>* :subgrp:*<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet -10`

Ottenere un ARN esistente in Amazon Neptune

Puoi ottenere l'ARN di una risorsa Neptune utilizzando l'API, AWS Management Console(AWS Command Line Interface)AWS CLI o Neptune.

Ottenere un ARN esistente utilizzando AWS Management Console

Per ottenere un ARN utilizzando la console, accedere alla risorsa per cui ottenere un ARN e visualizzane i relativi dettagli. Ad esempio, per ottenere l'ARN per un'istanza database, scegli Instances (Istanze) nel riquadro di navigazione e successivamente l'istanza desiderata dall'elenco. L'ARN è indicato nella sezione Instance Details (Dettagli istanza).

Ottenere un ARN esistente utilizzando AWS CLI

Per usare AWS CLI per ottenere un ARN per una particolare risorsa di Neptune, usa il comando per quella risorsa. `describe` La tabella seguente mostra ogni AWS CLI comando e la proprietà ARN utilizzata con il comando per ottenere un ARN.

AWS CLI Comando	Proprietà ARN
describe-event-subscriptions	EventSubscriptionArn
describe-certificates	CertificateArn
describe-db-parameter-groups	DB ParameterGroupArn
describe-db-cluster-parameter-gruppi	DB ClusterParameterGroupArn
describe-db-instances	DB InstanceArn

AWS CLI Comando	Proprietà ARN
describe-events	SourceArn
describe-db-subnet-groups	DB SubnetGroupArn
describe-db-clusters	DB ClusterArn
describe-db-cluster-snapshots	DB ClusterSnapshotArn

Ad esempio, il AWS CLI comando seguente ottiene l'ARN per un'istanza DB.

Example

Per Linux, OS X o Unix:

```
aws neptune describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2
```

Per Windows:

```
aws neptune describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2
```

Recupero di un ARN esistente utilizzando l'API

Per ottenere un ARN per una determinata risorsa Neptune, chiama le operazioni dell'API seguenti e usa le proprietà ARN mostrate.

Azione dell'API di Neptune	Proprietà ARN
DescribeEventSubscriptions	EventSubscriptionArn
DescribeCertificates	CertificateArn
Descrive B ParameterGroups	DB ParameterGroupArn

Azione dell'API di Neptune	Proprietà ARN
Descrivi B ClusterParameterGroups	DB ClusterParameterGroupArn
DescribeDBInstances	DB InstanceArn
DescribeEvents	SourceArn
Descrivi B SubnetGroups	DB SubnetGroupArn
DescribeDBClusters	DB ClusterArn
Descrivi B ClusterSnapshots	DB ClusterSnapshotArn

Backup e ripristino di un cluster database Amazon Neptune

Questa sezione illustra come effettuare il backup e il ripristino dei cluster database Amazon Neptune.

Argomenti

Panoramica di backup e ripristino di un cluster database Neptune

Questa sezione fornisce informazioni generali sul backup e il ripristino dei dati in Amazon Neptune.

Argomenti

Tolleranza ai guasti di un cluster database Neptune.

Un cluster database Neptune è tollerante ai guasti per impostazione predefinita. Il volume del cluster si estende su più zone di disponibilità in una singola regione AWS e ciascuna di esse include una copia dei dati del volume del cluster. Questa funzionalità consente al cluster DB di tollerare il guasto di una zona di disponibilità senza perdita di dati e con solo una breve interruzione del servizio.

In caso di errore dell'istanza primaria di un cluster database, Neptune esegue automaticamente il failover su una nuova istanza primaria, in uno dei due modi seguenti:

- Promuovendo una replica Neptune esistente come nuova istanza primaria
- Creando una nuova istanza primaria

Se il cluster database include una o più repliche di Neptune, una di queste verrà promossa come istanza primaria durante un evento di errore. Un evento di errore ha come conseguenza una breve interruzione, durante la quale le operazioni di lettura e scrittura inviate all'istanza primaria falliscono con un'eccezione. Tuttavia, il servizio viene in genere ripristinato in meno di 120 secondi e spesso in meno di 60 secondi. Per aumentare la disponibilità del cluster database, è consigliabile di creare almeno una o più repliche di Neptune in due o più due zone di disponibilità.

Puoi personalizzare l'ordine di promozione a istanza primaria delle repliche di Neptune dopo un errore assegnando una priorità a ciascuna di esse. Le priorità vanno da 0, quella massima, a 15, quella minima. Se si verifica un errore nell'istanza primaria, Neptune promuoverà a istanza primaria la replica di con la massima priorità. Puoi modificare la priorità di una replica di Neptune in qualsiasi momento. Modificando una priorità non attiverai un failover.

Puoi usare la AWS CLI per impostare la priorità di failover di un'istanza database, nel modo seguente:

```
aws neptune modify-db-instance --db-instance-identifier (the instance ID) --promotion-
tier (the failover priority value)
```

Più repliche di Neptune possono condividere la stessa priorità, generando così livelli di promozione. Se due o più repliche di Neptune condividono la stessa priorità, Neptune promuoverà quella di dimensioni maggiori. Se due o più repliche di Neptune condividono la stessa priorità e le stesse dimensioni, Neptune ne promuove una arbitrariamente nello stesso livello di promozione.

Se il cluster database non contiene repliche di Neptune, l'istanza primaria viene ricreata durante un evento di errore. Un evento di errore ha come conseguenza un'interruzione, durante la quale le operazioni di lettura e scrittura inviate all'istanza primaria falliscono con un'eccezione. Il servizio viene ripristinato quando crei la nuova istanza primaria. In genere, ciò richiede meno di 10 minuti. La promozione di una replica di Neptune a istanza primaria è un'operazione molto più veloce rispetto alla creazione di una nuova istanza primaria.

Backup di Neptune

Neptune esegue automaticamente il backup del volume del cluster e conserva i dati di ripristino per la durata del periodo di conservazione del backup. I backup di Neptune sono continui e incrementali, in modo da poter eseguire rapidamente un ripristino a qualsiasi momento nel tempo del periodo di conservazione del backup. Durante la scrittura dei dati di backup, non si verifica alcun impatto sulle prestazioni o interruzione del funzionamento del servizio del database. Puoi specificare un tempo di conservazione del backup, da 1 a 35 giorni, durante la creazione o la modifica di un cluster di database.

Per controllare l'utilizzo dello storage di backup, è possibile ridurre l'intervallo di retention dei backup e/o rimuovere gli snapshot manuali obsoleti e non più necessari. Per controllare i costi, è sufficiente monitorare la quantità di spazio di archiviazione occupata da backup continui e snapshot manuali che persistono oltre il periodo di conservazione e, all'occorrenza, ridurre l'intervallo di conservazione dei backup nonché rimuovere gli snapshot manuali non più necessari.

Se desideri conservare un backup oltre il tempo di conservazione del backup, puoi anche eseguire uno snapshot dei dati del volume del cluster. All'archiviazione degli snapshot verranno applicati i costi di archiviazione standard per Neptune. Per ulteriori informazioni sui prezzi di archiviazione di Neptune, consulta Prezzi di Amazon Neptune.

Neptune conserva i dati di ripristino incrementali per l'intero periodo di conservazione dei backup. Quindi, è sufficiente creare uno snapshot per i dati che desideri conservare oltre il periodo di conservazione dei backup. Puoi creare un nuovo cluster database dallo snapshot.

> **Important**
>
> Se si elimina un cluster database, tutti i backup automatici al suo interno vengono contemporaneamente eliminati e non possono essere ripristinati. Di conseguenza, a meno che tu non scelga di creare manualmente uno snapshot di database finale, non potrai ripristinare lo stato finale dell'istanza database in un secondo momento. Gli snapshot manuali non vengono eliminati quando viene eliminato il cluster.

> **Note**
>
> - Per i cluster database Amazon Neptune, il periodo di conservazione dei backup predefinito è di un giorno indipendentemente dal modo in cui viene creato il cluster database.
> - Non è possibile disabilitare i backup automatici in Neptune. Il periodo di conservazione dei backup per Neptune viene gestito dal cluster database.

Metriche di CloudWatch utili per la gestione dell'archiviazione di backup di Neptune

Per verificare e monitorare la quantità di spazio di archiviazione utilizzata dai backup di Neptune, è possibile usare le metriche di Amazon CloudWatch `TotalBackupStorageBilled`, `SnapshotStorageUsed` e `BackupRetentionPeriodStorageUsed`, nel seguente modo:

- `BackupRetentionPeriodStorageUsed` rappresenta la quantità di storage di backup, in byte, al momento utilizzata per l'archiviazione di backup continui. Questo valore dipende dalle dimensioni del volume del cluster e dalla quantità di modifiche apportate durante il periodo di retention. Tuttavia, ai fini della fatturazione, non supera mai il volume di cluster cumulativo nel periodo di retention. Ad esempio, se la dimensione del cluster `VolumeBytesUsed` è 107,374,182,400 byte (100 GiB) e il periodo di retention è pari a due giorni, il valore massimo per `BackupRetentionPeriodStorageUsed` è 214,748,364,800 byte (100 GiB + 100 GiB).

- `SnapshotStorageUsed` rappresenta la quantità di storage di backup utilizzata, in byte, per l'archiviazione di snapshot manuali oltre il periodo di retention dei backup. Gli snapshot manuali non vengono conteggiati sullo storage di backup dello snapshot mentre il timestamp di creazione è entro il periodo di retention. Anche tutti gli snapshot automatici non vengono inclusi nello storage di backup degli snapshot. Ogni snapshot corrisponde, per dimensioni, al volume del cluster al momento della sua acquisizione. Il valore di `SnapshotStorageUsed` dipende dal numero e dalle dimensioni degli snapshot conservati. Supponiamo, ad esempio, di disporre di uno snapshot manuale al di fuori del periodo di retention, con una dimensione del cluster `VolumeBytesUsed`, al momento dell'acquisizione di tale snapshot, di 100 GiB. Il valore di SnapshotStorageUsed sarà pari a 107,374,182,400 byte (100 GiB).
- `TotalBackupStorageBilled` corrisponde alla somma, in byte, di `BackupRetentionPeriodStorageUsed` e `SnapshotStorageUsed`, meno una quantità di storage di backup gratuito che è uguale alla dimensione del volume del cluster per un giorno. Lo storage di backup gratuito è uguale alla dimensione del volume più recente. Ad esempio, se la dimensione del cluster `VolumeBytesUsed` è 100 GiB, il periodo di retention è di due giorni e se si dispone di uno snapshot manuale al di fuori del periodo di retention, `TotalBackupStorageBilled` è 214,748,364,800 byte (200 GiB + 100 GiB - 100 GiB).

È possibile monitorare i cluster Neptune ed elaborare report con le metriche di CloudWatch tramite la console CloudWatch. Per ulteriori informazioni su come utilizzare i parametri CloudWatch, consulta Monitoraggio di Neptune e la tabella dei parametri in Metriche di Neptune CloudWatch .

Ripristino dei dati da un backup di Neptune

Puoi recuperare i dati creando un nuovo cluster database Neptune dai dati di backup conservati da Neptune o da uno snapshot del cluster database salvato. Puoi ripristinare rapidamente una nuova copia di un cluster DB creato dai dati di backup a un momento qualsiasi del periodo di retention dei backup. Data la natura continua e incrementale dei backup di Neptune durante il periodo di conservazione dei backup, non dovrai acquisire snapshot frequenti dei dati per migliorare i tempi di ripristino.

Per determinare l'ora ripristinabile più recente o meno recente di un'istanza database, individua i valori `Latest Restorable Time` o `Earliest Restorable Time` nella console Neptune. L'ultima ora ripristinabile di un cluster DB corrisponde al punto più recente in corrispondenza del quale puoi ripristinare il cluster DB, ovvero generalmente entro 5 minuti dall'ora corrente. La prima ora ripristinabile indica il punto all'interno del periodo di retention dei backup dai cui puoi ripristinare il volume del cluster.

Puoi stabilire il completamento del ripristino di un cluster DB verificando i valori `Latest Restorable Time` ed `Earliest Restorable Time`. I valori `Latest Restorable Time` ed `Earliest Restorable Time` restituiranno NULL fino all'avvenuto completamento dell'operazione di ripristino. Non puoi richiedere l'esecuzione di un'operazione di backup o ripristino se `Latest Restorable Time` o `Earliest Restorable Time` restituisce NULL.

Per eseguire il ripristino di un'istanza database a un'ora specifica utilizzando la AWS Management Console

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, seleziona Instances (Istanze). Scegliere l'istanza principale del cluster di database da ripristinare.
3. Scegliere Instance actions (Operazioni istanza), quindi Restore to point in time (Ripristina al point-in-time).

   Nella finestra Launch DB Instance (Avvia istanza database), scegliere Custom (Personalizza) in Restore time (Ora ripristino).
4. Specificare la data e l'ora del ripristino in Custom (Personalizza).
5. Immettere il nome della nuova istanza database ripristinata per DB instance identifier (Identificatore istanze DB) in Settings (Impostazioni).
6. Scegliere Launch DB Instance (Avvia istanza database) per avviare l'istanza database ripristinata.

   Verrà creata una nuova istanza database con il nome specificato e verrà inoltre creato un nuovo cluster DB. Il nome del cluster DB sarà costituito dal nome della nuova istanza database seguito da `-cluster`. Ad esempio, se il nome della nuova istanza database è `myrestoreddb`, il nome del nuovo cluster DB sarà `myrestoreddb-cluster`.

Finestra di backup in Neptune

I backup automatici vengono effettuati quotidianamente durante la finestra di backup scelta. Se il backup richiede più tempo rispetto alla finestra di backup prevista, il backup continua dopo il termine della finestra, finché non viene completato. La finestra di backup non può sovrapporsi con la finestra di manutenzione settimanale per l'istanza database.

Durante la finestra di backup automatico, le operazioni I/O di storage potrebbero essere sospese brevemente durante l'inizializzazione del processo di backup (in genere per alcuni secondi). Potresti rilevare un aumento della latenza per alcuni minuti durante i backup per le implementazioni Multi-AZ.

La finestra di backup viene normalmente selezionata in modo casuale da un blocco di otto ore per regione dal piano di controllo Amazon RDS sottostante di Neptune. I blocchi di ore per ogni regione da cui vengono assegnate le finestre di backup predefinite sono documentati nella sezione Finestra di backup della Guida per l'utente di Amazon RDS.

Creazione di uno snapshot del cluster database in Neptune

Neptune crea uno snapshot dei volumi di archiviazione del cluster database eseguendo il backup dell'intero cluster anziché dei singoli database. Quando crei uno snapshot di un cluster database, devi identificare il cluster database di cui eseguire il backup. Assegna quindi un nome allo snapshot del cluster database per poterlo ripristinare in un secondo momento. La quantità di tempo necessaria per creare uno snapshot del cluster database varia a seconda delle dimensioni dei database. Lo snapshot include l'intero volume di archiviazione. Di conseguenza, anche le dimensioni dei file, ad esempio i file temporanei, influiscono sulla quantità di tempo necessaria per creare lo snapshot.

Puoi creare uno snapshot del cluster database tramite la AWS Management Console, la AWS CLI o l'API di Neptune.

Utilizzo della console per creare uno snapshot del cluster database

Per creare uno snapshot del cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, scegli Database.
3. Seleziona l'istanza primaria per il cluster database nell'elenco delle istanze database.
4. Scegli Operazioni istanza, quindi Acquisisci snapshot.

   Viene visualizzata la finestra Take DB Snapshot (Acquisisci snapshot DB).
5. Nella casella Snapshot name (Nome snapshot), immettere il nome dello snapshot del cluster di database.
6. Scegli Take Snapshot (Acquisisci snapshot).

Ripristino da una snapshot del cluster di database

Quando crei uno snapshot Amazon Neptune di un cluster database, Neptune crea uno snapshot dei volumi di archiviazione del cluster eseguendo il backup di tutti i dati anziché delle singole istanze. È possibile creare un nuovo cluster database ripristinandolo da questo snapshot. Quando ripristini il cluster database, devi specificare il nome dello snapshot del cluster database da cui effettuare il ripristino e successivamente il nome per il nuovo cluster database creato dal ripristino.

Indice

Aspetti da considerare per il ripristino di un cluster database Neptune da uno snapshot

Non è possibile eseguire il ripristino da un cluster database esistente.

Il processo di ripristino crea sempre un nuovo cluster database, quindi non è possibile eseguire il ripristino su un cluster database già esistente.

Non viene ripristinata alcuna istanza

A un nuovo cluster database creato da un ripristino non sono associate istanze.

Una volta completato il ripristino e reso disponibile il nuovo cluster database, è necessario creare esplicitamente le istanze necessarie. È possibile effettuare tale operazione nella console Neptune o tramite l'API CreateDBInstance.

Non viene ripristinato alcun gruppo di parametri personalizzato

A un nuovo cluster database creato da un ripristino viene automaticamente associato il gruppo di parametri di database predefinito.

Una volta completato il ripristino e reso disponibile il nuovo cluster database, occorre associare tutti i gruppi di parametri di database personalizzati utilizzati dall'istanza da cui è stato eseguito il ripristino. A tale scopo, usa il comando Modifica sulla console Neptune o l'API ModifyDBInstance.

> **Important**
>
> È consigliabile salvare un gruppo di parametri personalizzato in uso nel cluster database di cui stai creando lo snapshot. In questo modo, dopo il ripristino da quello snapshot, sarà possibile associare facilmente il gruppo di parametri corretto al cluster database ripristinato.

Non viene ripristinato alcun gruppo di sicurezza personalizzato

A un nuovo cluster database creato da un ripristino viene automaticamente associato il gruppo di sicurezza predefinito.

Una volta completato il ripristino e reso disponibile il nuovo cluster database, occorre associare tutti i gruppi di sicurezza personalizzati utilizzati dall'istanza da cui è stato eseguito il ripristino. A tale scopo, usa il comando Modifica sulla console Neptune o l'API ModifyDBInstance.

Non è possibile ripristinare un cluster da uno snapshot condiviso e crittografato.

Non puoi ripristinare un cluster database da uno snapshot del cluster database che è sia condiviso che crittografato.

Puoi invece creare una copia dello snapshot non condivisa e ripristinare il cluster dalla copia.

Un cluster database ripristinato utilizza lo stesso spazio di archiviazione di quello precedente

Quando si ripristina un cluster DB da uno snapshot del cluster DB, la quantità di archiviazione allocata al nuovo cluster è la stessa allocata al cluster DB da cui è stata creato lo snapshot, indipendentemente dalla quantità di archiviazione allocata effettivamente utilizzata.

In altre parole, il limite massimo che ti viene fatturato non subisce modifiche. La reimpostazione del limite massimo richiede l'esportazione dei dati dal grafo e quindi il ricaricamento su un nuovo cluster DB (consulta Fatturazione dell'archiviazione Neptune).

Come eseguire il ripristino da uno snapshot

È possibile ripristinare un cluster database da uno snapshot del cluster database tramite la AWS Management Console, la AWS CLI o l'API Neptune.

Utilizzo della console per ripristinare da uno snapshot

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Scegliere lo snapshot cluster database da cui eseguire il ripristino.
4. Scegliere Actions (Operazioni), Restore Snapshot (Ripristina snapshot).
5. Nella pagina Restore DB Instance (Ripristina istanza database), nella casella DB Instance Identifier (Identificatore istanze database), digitare il nome del cluster di database ripristinato.
6. Selezionare Restore DB Instance (Ripristina istanza database).
7. Per ripristinare la funzionalità del cluster di database a quella del cluster da cui è stata creata la snapshot, occorre modificare il cluster di database in modo che utilizzi il gruppo di sicurezza. La fase successiva presume che il cluster di database si trovi in un cloud privato virtuale (VPC, Virtual Private Cloud). Se il cluster database non risiede in un VPC, utilizza la console Amazon EC2 per individuare il gruppo di sicurezza necessario per il cluster database.

    a. Accedi alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
    b. Fai clic su Security Groups (Gruppi di sicurezza) nel riquadro di navigazione.
    c. Seleziona il gruppo di sicurezza da utilizzare per i cluster di database. Se necessario, aggiungere regole per collegare il gruppo di sicurezza a un gruppo di sicurezza per un'istanza EC2.

Copia di una snapshot cluster database

Con Neptune puoi copiare snapshot del cluster database automatici o manuali. Dopo aver copiato uno snapshot, la copia è uno snapshot manuale.

È possibile copiare uno snapshot all'interno della stessa regione AWS e tra regioni AWS.

Per copiare uno snapshot automatica in un altro account AWS è prima necessario creare uno snapshot manuale dallo snapshot automatico e quindi copiare lo snapshot manuale nell'altro account.

Come alternativa alla copia, puoi anche condividere snapshot manuali con altri account AWS. Per ulteriori informazioni, consulta Condivisione di uno snapshot cluster database.

Argomenti

Limitazioni per la copia di una snapshot

Di seguito sono riportate alcune limitazioni che si applicano quando si copiano le snapshot:

- Puoi copiare uno snapshot tra Cina (Pechino) e Cina (Ningxia), ma non puoi copiare uno snapshot tra queste regioni della Cina e altre regioni AWS.
- Puoi copiare uno snapshot tra AWS GovCloud (US-East) e AWS GovCloud (US-West), ma non puoi copiare uno snapshot tra queste regioni AWS GovCloud (US) e altre regioni AWS.
- Se elimini una snapshot origine prima che la snapshot target diventi disponibile, la copia della snapshot potrebbe non riuscire. Verifica che la snapshot target abbia lo stato di `AVAILABLE` prima di eliminare una snapshot origine.
- Per ogni account è consentito un massimo di 5 richieste di copia di snapshot in corso in una singola regione.
- A seconda delle regioni coinvolte e la quantità di dati da copiare, la copia di una snapshot tra regioni potrebbe richiedere diverse ore.

In presenza di un numero elevato di richieste di copia snapshot tra regioni da una regione AWS di origine specifica, Neptune può inserire le nuove richieste di copia tra regioni da quella regione AWS di origine in una coda fino a quando non vengono completate alcune delle copie in corso. Nessuna informazione di avanzamento viene visualizzata relativamente alle richieste di copia mentre sono in coda. Le informazioni sull'avanzamento vengono visualizzate solo dopo l'avvio della copia.

Conservazione delle copie di snapshot relative a cluster di database

Neptune elimina gli snapshot automatici in questo modo:

- Al termine del periodo di conservazione.
- Quando si disabilitano le snapshot automatiche per un cluster di database.
- Quando si elimina un cluster di database.

Se vuoi mantenere uno snapshot automatico per un periodo più lungo, puoi copiarlo e creare uno snapshot manuale che sarà conservato fino a quando non lo elimini. Potrebbero essere addebitati costi di archiviazione di Neptune per gli snapshot manuali se questi superano lo spazio di archiviazione predefinito.

Per ulteriori informazioni sui costi dell'archiviazione di backup, consulta Prezzi di Neptune.

Gestione della crittografia durante la copia di snapshot

Puoi copiare una snapshot che è stata crittografata utilizzando una chiave di crittografia AWS KMS. Se la copia di una snapshot crittografata, la copia della snapshot deve anche essere crittografata. È possibile crittografare la copia con la stessa chiave di crittografia AWS KMS dello snapshot originale oppure specificare un'altra chiave di crittografia AWS KMS.

Non è possibile crittografare uno snapshot del cluster di database non crittografato durante la sua copia.

Per gli snapshot del cluster database Amazon Neptune, esiste anche l'opzione di lasciare lo snapshot del cluster database non crittografato e invece specificare una chiave di crittografia AWS KMS durante il ripristino. Il cluster DB ripristinato viene crittografato utilizzando la chiave specifica.

Copia di snapshot tra regioni AWS

> **Note**
>
> Questa funzionalità è disponibile a partire dal rilascio 1.0.2.1 del motore Neptune.

Quando copi uno snapshot in una regione AWS diversa dalla regione AWS dello snapshot di origine, la prima copia è una copia snapshot completa, anche se copi uno snapshot incrementale. Una copia snapshot completa contiene tutti i dati e i metadati necessari per archiviare l'istanza database. Dopo la copia del primo snapshot, puoi copiare snapshot incrementali della stessa istanza database nella stessa regione di destinazione all'interno dello stesso account AWS.

Una snapshot incrementale contiene solo i dati modificati dopo la snapshot più recente della stessa istanza database. La copia di snapshot incrementali è più rapida e comporta costi di archiviazione inferiori rispetto alla copia di snapshot complete. La copia di snapshot incrementali tra regioni AWS è supportata per snapshot crittografati e non crittografati.

> **Important**
>
> Per gli snapshot condivisi, la copia di snapshot incrementale non è supportata. Per gli snapshot condivisi, tutte le copie sono snapshot completi, anche all'interno della stessa regione.

A seconda delle regioni AWS coinvolte e della quantità di dati da copiare, la copia di uno snapshot tra regioni potrebbe richiedere diverse ore.

Copia di una snapshot di cluster di database tramite la console

Se il motore di database di origine è Neptune, lo snapshot è uno snapshot del cluster database. Per ogni account AWS, puoi copiare fino a cinque snapshot di cluster database alla volta per regione AWS. Viene supportata la copia di snapshot cluster database crittografate e non crittografate.

Per ulteriori informazioni sui prezzi del trasferimento dati, consulta Prezzi di Neptune.

Per cancellare un'operazione di copia quando è in corso, elimina la snapshot di cluster di database target mentre la snapshot di cluster è nello stato copying (copia in corso).

La seguente procedura funziona per la copia di snapshot di cluster di database crittografate e non:

Per copiare una snapshot cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Seleziona la casella di controllo per lo snapshot cluster database che vuoi copiare.
4. Scegli Actions (Operazioni), quindi Copy Snapshot (Copia snapshot). Viene visualizzata la pagina Make Copy of DB Snapshot (Crea copia di snapshot DB).
5. Digita il nome della copia di snapshot di cluster di database in New DB Snapshot Identifier (Nuovo identificatore snapshot DB).
6. Per copiare i tag e i valori dalla snapshot alla copia della snapshot, seleziona Copy Tags (Copia tag).
7. Per Enable Encryption (Abilita crittografia), seleziona una delle seguenti opzioni:
   - Seleziona Disable encryption (Disabilita crittografia) se lo snapshot cluster database non è crittografato e non vuoi crittografare la copia.
   - Scegli Enable encryption (Abilita crittografia) se lo snapshot cluster database non è crittografato, ma vuoi crittografare la copia. In questo caso, per Chiave master, specifica l'identificatore della chiave AWS KMS da utilizzare per crittografare la copia snapshot del cluster database.
   - Scegli Enable encryption (Abilita crittografia) se lo snapshot cluster database è crittografato. In questo caso, devi crittografare la copia, quindi l'opzione Yes (Sì) è già selezionata. Per Chiave master, specifica l'identificatore della chiave AWS KMS da utilizzare per crittografare la copia snapshot del cluster database.
8. Seleziona Copy Snapshot (Copia snapshot).

Copia di uno snapshot di cluster DB tramite l'AWS CLI

Puoi copiare uno snapshot di database tramite il comando copy-db-cluster-snapshot di AWS CLI.

Se stai copiando lo snapshot in una nuova regione AWS, esegui il comando nella nuova regione.

Utilizzare le descrizioni e gli esempi di parametri seguenti per determinare quali parametri utilizzare nella copia di uno snapshot con l'AWS CLI.

- `--source-db-cluster-snapshot-identifier` – Identificatore per lo snapshot DB origine.

- Se lo snapshot di origine si trova nella stessa regione AWS della copia, specifica un'identificatore dello snapshot di database valido, ad esempio `neptune:instance1-snapshot-20130805`.
- Se lo snapshot di origine si trova in una regione AWS diversa rispetto alla copia, specifica un nome ARN per lo snapshot di database valido, ad esempio `arn:aws:neptune:us-west-2:123456789012:snapshot:instance1-snapshot-20130805`.
- Se stai copiando da uno snapshot DB manuale condiviso, questo parametro deve essere l'Amazon Resource Name (ARN) della snapshot DB condivisa.
- Se stai copiando uno snapshot crittografato, questo parametro deve essere nel formato ARN per la regione AWS di origine e deve corrispondere a `SourceDBSnapshotIdentifier` nel parametro `PreSignedUrl`.

- `--target-db-cluster-snapshot-identifier` – – Identificatore per la nuova copia dello snapshot di database crittografato.
- `--kms-key-id` – – ID della chiave AWS KMS per uno snapshot di database crittografato. L'ID della chiave AWS KMS è il nome della risorsa Amazon (ARN), l'identificatore della chiave AWS KMS o l'alias della chiave AWS KMS per la chiave di crittografia AWS KMS.
  - Se copi uno snapshot di database crittografato dall'account AWS, puoi specificare un valore per questo parametro per crittografare la copia con una nuova chiave di crittografia AWS KMS. Se non specifichi un valore per questo parametro, la copia dello snapshot di database viene crittografata con la stessa chiave AWS KMS dello snapshot di database di origine.
  - Non puoi utilizzare questo parametro per creare una copia crittografata di uno snapshot non crittografato. Se tenti di eseguire questa operazione, verrà generato un errore.
  - Se copi uno snapshot crittografato in una regione AWS diversa, devi specificare una chiave AWS KMS per la regione AWS di destinazione. Le chiavi di crittografia AWS sono specifiche per la regione AWS KMS in cui vengono create e non puoi utilizzare le chiavi di una regione AWS in un'altra regione AWS.
- `--source-region` – – ID della regione AWS in cui risiede lo snapshot di database di origine. Se copi uno snapshot crittografato in una regione AWS diversa, allora dovrai specificare questa opzione.
- `--region` – – ID della regione AWS in cui viene copiato lo snapshot. Se copi uno snapshot crittografato in una regione AWS diversa, allora dovrai specificare questa opzione.

Example Da non crittografata, alla stessa regione

Il codice seguente crea una copia di uno snapshot, con il nuovo nome `mydbsnapshotcopy`, dalla regione AWS `us-east-1` alla regione `us-west-2`.

Per Linux, OS X o Unix:

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

Per Windows:

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

Example Da non crittografata, tra regioni

Il codice seguente crea una copia di uno snapshot, con il nuovo nome `mydbsnapshotcopy`, dalla regione AWS `us-east-1` alla regione `us-west-2`. Esegui il comando nella regione `us-west-2`.

Per Linux, OS X o Unix:

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
east-1:123456789012:snapshot:instance1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy \
  --source-region us-east-1 \
  --region us-west-2
```

Per Windows:

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
east-1:123456789012:snapshot:instance1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy ^
  --source-region us-east-1 ^
  --region us-west-2
```

Example Da crittografata, tra regioni

Nell'esempio di codice seguente viene copiato uno snapshot di database crittografato dalla regione AWS `us-east-1` alla regione `us-west-2`. Esegui il comando nella regione `us-west-2`.

Per Linux, OS X o Unix:

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
west-2:123456789012:snapshot:instance1-snapshot-20161115 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy \
  --source-region us-east-1 \
  --region us-west-2
  --kms-key-id my_us_west_2_key
```

Per Windows:

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
west-2:123456789012:snapshot:instance1-snapshot-20161115 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy ^
  --source-region us-east-1 ^
  --region us-west-2
  --kms-key-id my-us-west-2-key
```

Condivisione di uno snapshot cluster database

Con Neptune puoi condividere uno snapshot del cluster database manuale nei modi seguenti:

- La condivisione di uno snapshot del cluster di database manuale, crittografato o non crittografato permette agli account AWS autorizzati di copiare lo snapshot.
- La condivisione di uno snapshot cluster database manuale, crittografato o non crittografato, permette agli account AWS autorizzati di ripristinare direttamente un cluster database dallo snapshot, anziché eseguirne una copia da cui effettuare il ripristino.

> **Note**
>
> Per condividere uno snapshot cluster database automatico, crea uno snapshot cluster database manuale copiando lo snapshot automatico e quindi condividi la copia.

Per ulteriori informazioni sul ripristino di un cluster di database da una snapshot di cluster di database, consulta Come eseguire il ripristino da uno snapshot.

Puoi condividere una snapshot manuale con al massimo altri 20 account AWS. Puoi anche condividere uno snapshot manuale non crittografato come pubblico, per renderlo disponibile a tutti gli account AWS. Quando si condivide una snapshot come pubblica, occorre fare attenzione che non siano incluse informazioni personali.

> **Note**
>
> Quando ripristini un cluster database da uno snapshot condiviso usando AWS Command Line Interface (AWS CLI) o l'API di Neptune, devi specificare il nome della risorsa Amazon (ARN) dello snapshot condiviso come identificatore dello snapshot.

Argomenti

Condivisione di una snapshot di cluster di database crittografata

Puoi condividere snapshot di cluster di database che sono state crittografate in stato inattivo usando l'algoritmo di crittografia AES-256. Per ulteriori informazioni, consulta Crittografia delle risorse Neptune inattive. A tal fine, occorre procedere nel seguente modo:

1. Condividi la chiave di crittografia di AWS Key Management Service (AWS KMS) utilizzata per crittografare la snapshot con qualsiasi account con cui desideri accedere alla snapshot.

   Puoi condividere le chiavi di crittografia AWS KMS con un altro account AWS aggiungendolo alla policy della chiave KMS. Per informazioni dettagliate sull'aggiornamento della policy della chiave, consulta la sezione relativa alle policy delle chiavi nella Guida per sviluppatori AWS KMS. Per un esempio di creazione di una policy delle chiavi, consulta Creazione di una policy IAM per abilitare la copia di una snapshot crittografata più avanti in questo argomento.
2. Usa la AWS Management Console, la AWS CLI o l'API di Neptune per condividere lo snapshot crittografato con gli altri account.

Queste limitazioni si applicano alla condivisione delle snapshot crittografate:

- Non puoi condividere gli snapshot crittografati come pubblici.
- Non puoi condividere uno snapshot crittografato con la chiave di crittografia AWS KMS predefinita dell'account AWS che ha condiviso lo snapshot.

Concedere l'accesso a una chiave crittografica di AWS KMS

Per consentire a un altro account AWS di copiare uno snapshot del cluster database crittografato dal tuo account, l'account con cui condividi lo snapshot deve avere accesso alla chiave KMS che ha crittografato lo snapshot. Per consentire a un altro account AWS di accedere a una chiave AWS KMS, aggiorna la policy della chiave relativa alla chiave KMS con il nome ARN dell'account AWS con cui esegui la condivisione come `Principal` nella policy della chiave KMS. Successivamente, autorizza l'operazione `kms:CreateGrant`. Per le istruzioni generali, consulta Consentire agli utenti in altri account di utilizzare una chiave KMS nella Guida per gli sviluppatori di AWS Key Management Service.

Dopo aver concesso a un account AWS l'accesso alla chiave di crittografia KMS, per copiare lo snapshot crittografato l'account AWS dovrà creare un utente IAM, se non ne ha già uno. Le restrizioni di sicurezza di KMS non consentono l'utilizzo dell'identità di un account AWS root per questo scopo.

L'account AWS dovrà inoltre collegare una policy IAM a tale utente IAM per permettergli di copiare uno snapshot del cluster database crittografato usando la chiave KMS.

Nel seguente esempio di policy chiavi, l'utente `111122223333` è il proprietario della chiave di crittografia KMS, mentre l'utente `444455556666` è l'account con cui viene condivisa la chiave. Questa policy della chiave aggiornata offre all'account AWS l'accesso alla chiave KMS, includendo l'ARN per l'identità dell'account root di AWS per l'utente `444455556666` come `Principal` per la policy e consentendo l'operazione `kms:CreateGrant`.

```
{
  "Id=": "key-policy-1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid=": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/KeyUser",
        "arn:aws:iam::444455556666:root"
      ]},
      "Action": [
        "kms:CreateGrant",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid=": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/KeyUser",
        "arn:aws:iam::444455556666:root"
      ]},
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": "*",
```

```
      "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
    }
  ]
}
```

Creazione di una policy IAM per abilitare la copia di una snapshot crittografata

Dopo aver concesso l'accesso alla chiave KMS all'account AWS esterno, il proprietario dell'account può creare una policy che consenta a un utente IAM creato per l'account di copiare uno snapshot crittografato mediante tale chiave KMS.

L'esempio seguente mostra una policy che può essere collegata a un utente IAM per l'account AWS `444455556666`. Consente all'utente IAM di copiare uno snapshot condiviso dall'account AWS `111122223333` che è stato crittografato con la chiave KMS `c989c1dd-a3f2-4a5d-8d96-e793d082ab26` nella regione `us-west-2`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid=": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey",
                "kms:CreateGrant",
                "kms:RetireGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"]
        },
        {
            "Sid=": "AllowAttachmentOfPersistentResources",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
```

```
            "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"],
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

Per informazioni dettagliate sull'aggiornamento della policy della chiave, consulta la sezione relativa alle policy delle chiavi nella Guida per sviluppatori AWS Key Management Service.

Condivisione di uno snapshot cluster database

Puoi condividere uno snapshot di cluster database usando la AWS Management Console, la AWS CLI o l'API di Neptune.

Utilizzo della console per condividere uno snapshot di cluster database

Usando la console Neptune, puoi condividere uno snapshot del cluster database manuale con un massimo di 20 account AWS. Puoi anche interrompere la condivisione di uno snapshot manuale con uno o più account.

Per condividere uno snapshot di cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Scegliere lo snapshot manuale da condividere.
4. Scegliere Actions (Operazioni), Share Snapshot (Condividi snapshot).
5. Scegliere una delle opzioni seguenti per DB snapshot visibility (Visibilità snapshot DB).

    - Se l'origine non è crittografata, scegli Pubblico per consentire a tutti gli account AWS di ripristinare un cluster database dallo snapshot del cluster database manuale. In alternativa, scegli Privato per consentire solo agli account AWS specificati di ripristinare un cluster database dallo snapshot del cluster database manuale.

> **Warning**
>
> Se si imposta Visibilità snapshot DB su Pubblico, tutti gli account AWS possono ripristinare un cluster di database dallo snapshot del cluster di database manuale e accedere ai dati. Non condividere snapshot di cluster di database manuali che contengono informazioni private impostandoli come Public (Pubblico).

- Se l'origine è crittografata, l'opzione DB snapshot visibility (Visibilità snapshot DB) è impostata su Private (Privato), perché gli snapshot crittografati non possono essere condivisi come pubblici.

6. Per ID account AWS, immetti l'identificatore dell'account AWS a cui si vuole consentire il ripristino di un cluster del database dallo snapshot manuale. Quindi scegliere Add (Aggiungi). Ripetere questa operazione per includere altri identificatori di account AWS, per un massimo di 20 account AWS.

   In caso di errori durante l'aggiunta di un identificatore di account AWS all'elenco di account consentiti, è possibile eliminare l'account in questione dall'elenco selezionando Elimina a destra dell'identificatore dell'account AWS errato.
7. Dopo aver aggiunto gli identificatori per tutti gli account AWS a cui si vuole consentire il ripristino dello snapshot manuale, scegli Salva.

Per interrompere la condivisione di uno snapshot del cluster di database manuale con un account AWS

1. Apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Selezionare lo snapshot manuale di cui interrompere la condivisione.
4. Scegliere Actions (Operazioni) e quindi Share Snapshot (Condividi snapshot).
5. Per rimuovere l'autorizzazione per un account AWS, selezionare Elimina per l’identificatore account AWS per quell’account dall’elenco di account autorizzati.
6. Seleziona Salva.

Eliminazione di uno snapshot di Neptune

È possibile eliminare uno snapshot del database utilizzando la AWS Management Console, la AWS CLI o l'API di gestione di Neptune:

Eliminazione tramite la console

1. Accedi alla Console di gestione AWS e apri la console Amazon Neptune all'indirizzo https://console.aws.amazon.com/neptune/home.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Scegliere la snapshot DB da eliminare.
4. Per Actions (Operazioni), scegliere Delete Snapshot (Elimina snapshot).
5. Nella pagina di conferma, scegliere Delete (Elimina).

Eliminazione mediante l'AWS CLI

È inoltre possibile eliminare uno snapshot DB utilizzando il comando dell'AWS CLI delete_db_cluster_snapshot, utilizzando il parametro `--db-snapshot-identifier` per identificare lo snapshot che si desidera eliminare:

Per Linux, OS X o Unix:

```
aws neptune delete-db-cluster-snapshot \
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

Per Windows:

```
aws neptune delete-db-cluster-snapshot ^
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

Eliminazione tramite l'API di gestione di Neptune

È possibile utilizzare uno degli SDK per eliminare uno snapshot DB richiamando l'API DeleteDBClusterSnapshot e utilizzare i parametri `DBSnapshotIdentifier` per identificare lo snapshot DB da eliminare.

Best practice: come utilizzare Neptune al meglio

Di seguito sono riportate alcune raccomandazioni generali sull'utilizzo di Amazon Neptune. Usa queste informazioni come riferimento per trovare rapidamente le raccomandazioni per l'utilizzo di Amazon Neptune e ottimizzare le prestazioni.

Indice

Linee guida operative di base per Amazon Neptune

Di seguito sono illustrate le linee guida operative di base da seguire quando si usa Neptune.

- Informazioni sulle istanze database di Neptune per poterle dimensionare in modo appropriato in base ai requisiti di prestazioni e al caso d'uso. Per informazioni, consultare Cluster e istanze database di Amazon Neptune.
- Monitora l'utilizzo della CPU e della memoria. Ciò ti consente di sapere quando eseguire la migrazione a una classe di istanza database con una maggiore capacità di memoria o di CPU per ottenere le prestazioni di query richieste. Amazon CloudWatch può essere configurato per avvisarti quando i modelli di utilizzo cambiano o quando stai per raggiungere la capacità massima della distribuzione. In questo modo può aiutarti a mantenere le prestazioni del sistema e la disponibilità. Consultare Monitoraggio delle istanze e Monitoraggio di Neptune per i dettagli.

  Poiché Neptune dispone del proprio gestore di memoria, è normale osservare un utilizzo della memoria relativamente basso anche quando l'utilizzo della CPU è elevato. Il verificarsi di eccezioni di memoria esaurita durante l'esecuzione di query è un chiaro indicatore della necessità di aumentare la memoria liberabile.
- Abilita i backup automatici e imposta la finestra di backup su un orario opportuno.
- Prova il failover per l'istanza database per capire quanto tempo impiega il processo per il tuo caso d'uso. Inoltre, ciò contribuisce a garantire che l'applicazione che accede all'istanza database è in grado di connettersi automaticamente alla nuova istanza database dopo il failover.
- Se possibile, eseguire il client e il cluster Neptune nella stessa regione e VPC, poiché le connessioni tra regioni con il peering VPC possono introdurre ritardi nei tempi di risposta delle query. Per le risposte delle query in pochi millisecondi, è necessario mantenere il client e il cluster Neptune nella stessa regione e nello stesso VPC.
- Quando si crea un'istanza di replica di lettura, questa deve essere grande almeno quanto l'istanza di scrittura master. Ciò consente di mantenere sotto controllo il ritardo di replica ed evitare il riavvio della replica. Per informazioni, consultare Evitare classi di istanze diverse in un cluster.
- Prima di eseguire l'aggiornamento a una nuova versione principale del motore, è necessario assicurarsi di testare l'applicazione su di essa prima di procedere. A tale scopo, è possibile clonare il cluster database affinché il cluster clone esegua la nuova versione del motore e quindi testare l'applicazione sul clone.
- Per facilitare i failover, tutte le istanze dovrebbero idealmente avere le stesse dimensioni.

Argomenti

- Best practice di Amazon Neptune per la sicurezza
- Evitare classi di istanze diverse in un cluster

Best practice di Amazon Neptune per la sicurezza

Utilizza gli account AWS Identity and Access Management (IAM) per controllare l'accesso alle azioni dell'API di Neptune. Controlla le azioni per creare, modificare o eliminare risorse Neptune, come istanze database, gruppi di sicurezza, gruppi di opzioni o gruppi di parametri, e quelle per eseguire azioni amministrative comuni, come il backup e il ripristino di istanze database.

- Usa credenziali temporanee anziché credenziali persistenti quando possibile.
- Assegna un account IAM singolo a ogni utente che gestisce le risorse Amazon Relational Database Service (Amazon RDS). Non usare mai utenti root dell'account AWS per gestire le risorse Neptune. Crea un utente IAM per tutti, incluso te stesso.
- Assegna a ciascun utente il set minimo di autorizzazioni richieste per eseguire le proprie mansioni.
- Utilizza gruppi IAM per gestire in modo efficace le autorizzazioni per più utenti.
- Ruota periodicamente le credenziali IAM.

Per ulteriori informazioni sull'utilizzo di IAM per l'accesso alle risorse Neptune, consulta Sicurezza in Amazon Neptune. Per informazioni generali sull'utilizzo di IAM, consulta AWS Identity and Access Management e Best practice di IAM nella Guida per l'utente di IAM.

Evitare classi di istanze diverse in un cluster

Quando il cluster database contiene istanze di classi diverse, nel tempo possono verificarsi problemi. Il problema più comune è che un'istanza di lettura di piccole dimensioni può entrare in un ciclo di riavvii ripetuti a causa del ritardo di replica. Se un nodo di lettura ha una configurazione della classe

dell'istanza database più debole rispetto a quella di un'istanza database di scrittura, il volume delle modifiche può essere troppo grande perché il nodo di lettura riesca a stare al passo.

> **Important**
>
> Per evitare riavvii ripetuti causati dal ritardo di replica, configura il cluster database in modo che tutte le istanze abbiano la stessa classe di istanza (dimensioni).

Per osservare il ritardo tra l'istanza di scrittura (primaria) e le istanze di lettura nel cluster database, puoi usare la metrica `ClusterReplicaLag` in Amazon CloudWatch. La metrica `VolumeWriteIOPs` consente inoltre di rilevare picchi di attività di scrittura nel cluster che possono creare ritardi nella replica.

Evitare i riavvii ripetuti durante il caricamento in blocco

Se si verifica un ciclo di riavvii ripetuti delle repliche di lettura a causa del ritardo di replica durante un caricamento in blocco, è probabile che le repliche non riescano a stare al passo con l'istanza di scrittura nel cluster database.

Dimensionare le istanze di lettura in modo che siano più grandi dell'istanza di scrittura oppure rimuoverle temporaneamente durante il caricamento in blocco e quindi ricrearle al termine dell'operazione.

Abilitazione dell'indice OSGP in presenza di un numero elevato di predicati

Se il modello di dati contiene un elevato numero di predicati distinti (nella maggior parte dei casi più di mille), le prestazioni potrebbero risultare ridotte con costi operativi più elevati.

In tal caso, è possibile migliorare le prestazioni abilitando l'indice OSGP. Per informazioni, consultare Indice OSGP.

Evitare le transazioni di lunga durata, ove possibile

Le transazioni di lunga durata, di sola lettura o di lettura e scrittura, possono causare problemi imprevisti dei seguenti tipi:

Una transazione di lunga durata su un'istanza di lettura o su un'istanza di scrittura con scritture simultanee può comportare un accumulo significativo di diverse versioni dei dati. Questo può causare latenze più elevate per le query di lettura che filtrano gran parte dei risultati.

In alcuni casi, le versioni accumulate nelle ore possono limitare le nuove scritture.

Una transazione di lettura-scrittura di lunga durata con molte scritture può causare problemi anche in caso di riavvio dell'istanza. Se un'istanza viene riavviata dopo un evento di manutenzione o un arresto anomalo, tutte le scritture di cui non è stato eseguito il commit vengono ripristinate. Tali operazioni di annullamento in genere vengono eseguite in background e non impediscono il ripristino dell'istanza, ma qualsiasi nuova scrittura che sia in conflitto con le operazioni in fase di rollback ha esito negativo.

Se ad esempio la stessa query viene ripetuta dopo l'interruzione della connessione nell'esecuzione precedente, potrebbe avere esito negativo al riavvio dell'istanza.

Il tempo necessario per annullare le operazioni è proporzionale alle dimensioni delle modifiche coinvolte.

Best practice per l'utilizzo delle metriche di Neptune

Per identificare i problemi di prestazioni causati da risorse insufficienti e altri colli di bottiglia comuni, puoi monitorare le metriche disponibili per il cluster database Neptune.

Monitora regolarmente i parametri relativi alle prestazioni per raccogliere dati riguardo ai valori medi, massimi e minimi per vari intervalli di tempo. Ciò consente di identificare quando le prestazioni subiscono un calo. Utilizzando questi dati, puoi configurare allarmi di Amazon CloudWatch che ti avvisano quando vengono raggiunte soglie delle metriche specifiche.

Quando configuri un nuovo cluster di database e lo esegui con un carico di lavoro tipico, prova ad acquisire i valori medi, massimi e minimi di tutti i parametri relativi alle prestazioni a intervalli diversi (ad esempio, un'ora, 24 ore, una settimana, due settimane). Ciò ti permette di avere un quadro dei valori normali. Ciò aiuta anche a effettuare confronti delle attività durante le ore di punta e non di punta. Puoi quindi utilizzare queste informazioni per identificare quando le prestazioni scendono al di sotto dei livelli standard e impostare gli allarmi di conseguenza.

Consulta Monitoraggio di Neptune tramite Amazon CloudWatch per informazioni su come visualizzare le metriche di Neptune.

Di seguito sono elencati i parametri più importanti con cui iniziare:

- BufferCacheHitRatio: percentuale di richieste gestite dalla cache del buffer. I mancati riscontri della cache aggiungono una latenza significativa all'esecuzione delle query. Se il rapporto di riscontri

della cache è inferiore al 99,9% e la latenza è un problema per l'applicazione, valuta la possibilità di aggiornare il tipo di istanza per memorizzare nella cache più dati in memoria.

- Utilizzo della CPU: percentuale di capacità di elaborazione del computer utilizzata. Valori di utilizzo della CPU elevati possono essere appropriati, a seconda degli obiettivi relativi alle prestazioni di query.
- Memoria liberabile: quantità di RAM disponibile nell'istanza database, in megabyte. Neptune dispone di un proprio gestore di memoria, quindi questa metrica potrebbe essere minore del previsto. Se le query generano spesso eccezioni di memoria esaurita, potrebbe essere necessario aggiornare la classe di istanza a una con una maggiore quantità di RAM.

Nella scheda Monitoring (Monitoraggio), la linea rossa indica un livello del 75% per i parametri CPU e memoria. Se il consumo di memoria dell'istanza supera spesso questa linea, controlla il carico di lavoro e valuta la possibilità di aggiornare l'istanza per migliorare le prestazioni di query.

Best practice per l'ottimizzazione delle query di Neptune

Uno dei modi migliori per migliorare le prestazioni di Neptune consiste nell'ottimizzare le query più comuni e a uso intensivo di risorse per rendere l'esecuzione meno costosa.

Per informazioni su come ottimizzare le query Gremlin, consulta Hint di query Gremlin e Ottimizzazione di query Gremlin. Per informazioni su come ottimizzare le query SPARQL, consulta Hint di query SPARQL.

Bilanciamento del carico tra le repliche di lettura

L'instradamento round robin dell'endpoint lettore funziona cambiando l'host a cui punta la voce DNS. Il client deve creare una nuova connessione e risolvere il record DNS per ottenere una connessione a una nuova replica di lettura, poiché le connessioni WebSocket vengono spesso mantenute in vita per lunghi periodi.

Per ottenere repliche di lettura diverse per richieste successive, assicurarsi che il client risolva la voce DNS ogni volta che si connette. Potrebbe essere necessario chiudere la connessione e riconnettersi all'endpoint del lettore.

É inoltre possibile eseguire il bilanciamento del carico delle richieste tra le repliche di lettura connettendo esplicitamente gli endpoint dell'istanza.

Caricamento più veloce con un'istanza temporanea di dimensioni più grandi

Le prestazioni di caricamento aumentano con le istanze di dimensioni più grandi. Se non utilizzi un tipo di istanza di grandi dimensioni, ma vuoi aumentare la velocità di caricamento, puoi utilizzare un'istanza più grande per caricare e poi eliminarla.

> **Note**
>
> La procedura seguente è per un nuovo cluster. Se hai già un cluster, puoi aggiungere una nuova istanza più grande e quindi promuoverla a istanza database primaria.

Per caricare i dati utilizzando un'istanza di dimensioni più grandi

1. Creare un cluster con una singola istanza `r5.12xlarge`. Questa è l'istanza database primaria.
2. Creare una o più repliche di lettura delle stesse dimensioni (`r5.12xlarge`).

   È possibile creare le repliche di lettura in dimensioni inferiori, ma se non sono abbastanza grandi per stare al passo con le scritture effettuate dall'istanza primaria, potrebbe essere necessario riavviarle di frequente. I tempi di inattività che ne derivano riducono drasticamente le prestazioni.
3. Nel comando dello strumento di caricamento in blocco, includi `“parallelism” : “OVERSUBSCRIBE”` per indicare a Neptune di utilizzare tutte le risorse della CPU disponibili per il caricamento (consulta Parametri della richiesta dello dello strumento di caricamento Neptune). L'operazione di caricamento procederà quindi alla velocità consentita dalle operazioni di I/O, che in genere richiedono il 60-70% delle risorse della CPU.
4. Caricare i dati utilizzando Neptune Loader. Il processo di caricamento viene eseguito nell'istanza database primaria.
5. Al termine del caricamento dei dati, assicurati di dimensionare tutte le istanze del cluster in base allo stesso tipo di istanza per evitare costi aggiuntivi e problemi di riavvii ripetuti (consulta Evitare istanze di dimensioni diverse).

Ridimensiona l'istanza di scrittura eseguendo il failover su una replica di lettura

Il modo migliore per ridimensionare un'istanza nel cluster database, inclusa l'istanza di scrittura, consiste nel creare o modificare un'istanza di replica di lettura affinché abbia le dimensioni desiderate

e quindi eseguire deliberatamente il failover su tale replica di lettura. Il tempo di inattività rilevato dall'applicazione corrisponde esclusivamente al tempo necessario per modificare l'indirizzo IP dell'istanza di scrittura, in genere compreso tra 3 e 5 secondi.

L'API di gestione di Neptune usata per eseguire deliberatamente il failover dell'istanza di scrittura corrente su un'istanza di replica di lettura è FailoverDBCluster. Se utilizzi il client Java Gremlin, potrebbe essere necessario creare un nuovo oggetto client dopo il failover per rilevare il nuovo indirizzo IP, come indicato qui.

Assicurati di modificare tutte le istanze impostandole sulle stesse dimensioni per evitare un ciclo di riavvii ripetuti, come indicato di seguito.

Nuovo tentativo di caricamento dopo l'errore di attività di prefetch dei dati interrotta

Quando si caricano dati in Neptune mediante lo strumento di caricamento in blocco, occasionalmente può essere restituito uno stato `LOAD_FAILED`, con un messaggio `PARSING_ERROR` e `Data prefetch task interrupted` segnalato in risposta a una richiesta di informazioni dettagliate, come segue:

```
"errorLogs" : [
  {
    "errorCode" : "PARSING_ERROR",
    "errorMessage" : "Data prefetch task interrupted: Data prefetch task for 11467
 failed",
    "fileName" : "s3://some-source-bucket/some-source-file",
    "recordNum" : 0
  }
]
```

Se si verifica questo errore, è sufficiente riprovare la richiesta di caricamento in blocco.

L'errore si verifica quando si è verificata un'interruzione temporanea che in genere non è stata causata dalla richiesta o dai dati e solitamente si può risolvere eseguendo nuovamente la richiesta di caricamento in blocco.

Se stai utilizzando impostazioni predefinite, ovvero `"mode":"AUTO"` e `"failOnError":"TRUE"`, lo strumento di caricamento salta i file che ha già caricato e riprende il caricamento dei file che non aveva ancora caricato quando si è verificata l'interruzione.

Best practice generali per l'utilizzo di Gremlin con Neptune

Segui queste raccomandazioni quando usi il linguaggio di attraversamento grafi Gremlin con Neptune. Per informazioni sull'uso di Gremlin con Neptune, consulta the section called “Gremlin”.

Important

Nella versione 3.4.11 di TinkerPop è stata apportata una modifica che migliora la correttezza della modalità di elaborazione delle query, ma per il momento l'impatto sulle prestazioni delle query può essere significativo.
Una query di questo tipo, ad esempio, può essere eseguita molto più lentamente:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  out()
```

I vertici dopo il passaggio limite vengono ora recuperati in modo non ottimale a causa della modifica introdotta in TinkerPop 3.4.11. Per evitare il problema, puoi modificare la query aggiungendo il passaggio barrier() in qualsiasi punto dopo `order().by()`. Ad esempio:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

TinkerPop 3.4.11 è stato abilitato nella versione 1.0.5.0 del motore Neptune.

Argomenti

- Uso di data e ora native per i dati dell'ora GLV

Test del codice Gremlin nel contesto in cui verrà implementato

In Gremlin, i client possono inviare richieste al server in diversi modi: tramite WebSocket o Bytecode GLV o mediante la console Gremlin usando script basati su stringhe.

È importante riconoscere che l'esecuzione delle query Gremlin può variare a seconda del modo in cui viene inviata la query. Una query che restituisce un risultato vuoto potrebbe essere considerata come riuscita se inviata in modalità Bytecode, ma come non riuscita se inviata in modalità script. Ad esempio, se si include `next()` in una query in modalità script, la funzione `next()` viene inviata al server, ma se si usa ByteCode la funzione `next()` viene in genere elaborata dal client stesso. Nel primo caso, la query ha esito negativo se non viene trovato alcun risultato, ma nel secondo, la query ha esito positivo indipendentemente dal fatto che il set di risultati sia vuoto o meno.

Se sviluppi e testi il codice in un contesto, ad esempio, la console Gremlin che generalmente invia le query in formato testo, ma poi implementi il codice in un contesto diverso, ad esempio tramite il driver Java che utilizza Bytecode, potrebbero verificarsi problemi, in quanto il comportamento del codice nell'ambiente di produzione risulta diverso da quello nell'ambiente di sviluppo.

> **Important**
>
> Assicurati di testare il codice Gremlin nel contesto GLV in cui verrà implementato, per evitare risultati imprevisti.

Strutturare le query di upsert per sfruttare il motore DFE

È possibile migliorare in modo significativo le prestazioni delle query di upsert sfruttando il motore Neptune DFE nel modo più completo possibile.

Creazione di upsert efficienti con i passaggi `mergeV()` e `mergeE()` di Gremlin spiega come strutturare le query di upsert per utilizzare il motore DFE nel modo più efficace possibile.

Creazione di scritture multithread Gremlin efficienti

Esistono alcune linee guida per il caricamento multithread di dati in Neptune con Gremlin.

Se possibile, fornisci a ciascun thread un set di vertici o archi da inserire o modificare che non siano in collisione. Ad esempio, il thread 1 per l'intervallo di ID 1–50.000, il thread 2 per l'intervallo di ID 50.001–100.000 e così via. In questo modo si riduce la possibilità di generare una `ConcurrentModificationException`. Per sicurezza, inserisci un blocco `try/catch` attorno a tutte le scritture. In caso di errore, è possibile riprovare dopo un po'.

In genere raggruppare le scritture in una dimensione batch compresa tra 50 e 100 (vertici o archi) si rivela una soluzione valida. Se per ciascun vertice viene aggiunto un numero elevato di proprietà, potrebbe essere preferibile un numero più vicino a 50 che a 100. Vale la pena di fare dei tentativi. Per le scritture in batch, è possibile utilizzare il codice seguente:

```
g.addV(‘test’).property(id,’1’).as(‘a’).
  addV(‘test’).property(id,’2’).
  addE(‘friend’).to(‘a’).
```

Tale codice viene quindi ripetuto in ciascuna operazione batch.

L'utilizzo di batch risulta significativamente più efficiente rispetto ad aggiungere un vertice o un arco per round trip Gremlin al server.

Se utilizzi un client Gremlin Language Variant (GLV), puoi creare un batch in modo programmatico creando prima un attraversamento. Quindi aggiungere ed eseguire l'iterazione su di esso, ad esempio:

```
  t.addV(‘test’).property(id,’1’).as(‘a’)
  t.addV(‘test’).property(id,’2’)
  t.addE(‘friend’).to(‘a’)
  t.iterate()
```

È consigliabile utilizzare il client Gremlin Language Variant, se possibile. Tuttavia, si può fare qualcosa di analogo con un client che inoltra le query come stringhe di testo concatenando le stringhe per creare un batch.

Se usi una delle librerie client Gremlin anziché il client HTTP di base per le query, i thread devono condividere tutti lo stesso client, cluster o pool di connessioni. Potrebbe essere necessario ottimizzare le impostazioni per ottenere la migliore velocità di trasmissione effettiva possibile, ad esempio le dimensioni del pool di connessioni e il numero di thread di lavoro utilizzati dal client Gremlin.

Eliminazione dei record con la proprietà dell'ora di creazione

Puoi eliminare i record obsoleti memorizzando l'ora di creazione come proprietà nei vertici e rilasciandoli periodicamente.

Se è necessario memorizzare i dati per una durata specifica e quindi rimuoverli dal grafo (time-to-live del vertice), puoi memorizzare una proprietà timestamp alla creazione del vertice. Quindi puoi eseguire periodicamente una query `drop()` per tutti i vertici creati prima di una determinata ora, ad esempio:

```
g.V().has(“timestamp”, lt(datetime('2018-10-11')))
```

Uso del metodo **`datetime( )`** per i dati dell'ora di Groovy

Neptune fornisce il metodo `datetime` per specificare la data e l'ora per le query inviate nella variante Gremlin Groovy. Ciò include la console Gremlin, le stringhe di testo che utilizzano REST API HTTP e qualsiasi altra serializzazione che utilizza Groovy.

> **Important**
>
> Si applica solo ai metodi dove invii la query Gremlin come stringa di testo. Se stai utilizzando Gremlin Language Variant (GLV), devi usare le classi e le funzioni di date native per la lingua. Per ulteriori informazioni, consulta la sezione successiva the section called “Data e ora in nativo”.
>
> A partire da TinkerPop `3.5.2` (introdotto nel rilascio 1.1.1.0 del motore Neptune), `datetime` è parte integrante di TinkerPop.

Puoi utilizzare il metodo `datetime` per memorizzare e confrontare le date:

```
g.V('3').property('date',datetime('2001-02-08'))
```

```
g.V().has('date',gt(datetime('2000-01-01')))
```

Uso di data e ora native per i dati dell'ora GLV

Se stai utilizzando Gremlin Language Variant (GLV), è necessario utilizzare le classi e le funzioni di data e ora native fornite dal linguaggio di programmazione per i dati relativi al tempo di Gremlin.

Le librerie ufficiali TinkerPop Java, Node.js (JavaScript), Python o .NET sono tutte librerie Gremlin Language Variant.

> **Important**
>
> Si applica solo alle librerie Gremlin Language Variant (GLV). Se utilizzi un metodo che prevede l'invio della query Gremlin come stringa di testo, è necessario utilizzare il metodo `datetime()` fornito da Neptune. Ciò include la console Gremlin, le stringhe di testo che utilizzano REST API HTTP e qualsiasi altra serializzazione che utilizza Groovy. Per maggiori informazioni consulta la sezione precedente, the section called “`datetime( )`”.

Python

Di seguito è riportato un esempio parziale in Python che crea una singola proprietà denominata "`date`" per il vertice con ID "`3`". Imposta il valore come data generata usando il metodo `datetime.now()` di Python.

```
import datetime

g.V('3').property('date',datetime.datetime.now()).next()
```

Per un esempio completo della connessione a Neptune utilizzando Python, consulta Utilizzo di Python per connettersi a un'istanza database Neptune

Node.js (JavaScript)

Di seguito è riportato un esempio parziale in JavaScript che crea una singola proprietà denominata "`date`" per il vertice con ID "`3`". Imposta il valore come data generata usando il costruttore `Date()` di Node.js.

```
g.V('3').property('date', new Date()).next()
```

Per un esempio completo della connessione a Neptune utilizzando Node.js, consulta Utilizzo di Node.js per connettersi a un'istanza database Neptune

Java

Di seguito è riportato un esempio parziale in Java che crea una singola proprietà denominata "`date`" per il vertice con ID "`3`". Imposta il valore come data generata usando il costruttore `Date()` di Java.

```
import java.util.date

g.V('3').property('date', new Date()).next();
```

Per un esempio completo della connessione a Neptune utilizzando Java, consulta Utilizzo di un client Java per connettersi a un'istanza database Neptune

.NET (C#)

Di seguito è riportato un esempio parziale in C# che crea una singola proprietà denominata "`date`" per il vertice con ID "`3`". Imposta il valore come data generata usando la proprietà *`DateTime.UtcNow`* di .NET.

```
Using System;

g.V('3').property('date', DateTime.UtcNow).next()
```

Per un esempio completo della connessione a Neptune utilizzando C#, consulta Utilizzo di .NET per connettersi a un'istanza database Neptune

Best practice per l'utilizzo del client Java Gremlin con Neptune

Utilizzo della versione più recente compatibile del client Java Apache TinkerPop

Se possibile, usa sempre la versione più recente del client Java Gremlin Apache TinkerPop supportata dalla versione del motore in uso. Le versioni più recenti contengono numerose correzioni di bug che migliorano la stabilità, le prestazioni e l'usabilità del client.

Per un elenco delle versioni del client compatibili con diverse versioni del motore Neptune, consulta Client Apache Java Gremlin TinkerPop .

Riutilizzo dell'oggetto client tra più thread

Riutilizza lo stesso oggetto client (o `GraphTraversalSource`) tra più thread. Ovvero, crea un'istanza condivisa di una classe `org.apache.tinkerpop.gremlin.driver.Client`

nell'applicazione anziché in ogni thread. L'oggetto `Client` è thread-safe e l'overhead di inizializzazione è notevole.

Questo vale anche per `GraphTraversalSource`, che crea un oggetto `Client` internamente. Ad esempio, il codice seguente determina la creazione di istanze di un nuovo oggetto `Client`:

```
import static
 org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource.traversal;

  /////

GraphTraversalSource traversal = traversal()
                                    .withRemote(DriverRemoteConnection.using(cluster));
```

Creazione di oggetti client Java Gremlin separati per gli endpoint di lettura e scrittura

Puoi aumentare le prestazioni eseguendo le scritture solo sull'endpoint writer e leggendo da uno o più endpoint di sola lettura.

```
Client readerClient = Cluster.build("https://reader-endpoint")
          ...
          .connect()

Client writerClient = Cluster.build("https://writer-endpoint")
          ...
          .connect()
```

Aggiunta di più endpoint di replica di lettura a un pool di connessioni Java Gremlin

Quando crei un oggetto `Cluster` di Java Gremlin, puoi utilizzare il metodo `.addContactPoint()` per aggiungere più istanze di replica di lettura ai punti di contatto del pool di connessioni.

```
Cluster.Builder readerBuilder = Cluster.build()
          .port(8182)
          .minConnectionPoolSize(…)
          .maxConnectionPoolSize(…)
          ……
```

```
            .addContactPoint("reader-endpoint-1")
            .addContactPoint("reader-endpoint-2")
```

Chiusura del client per evitare il limite di connessioni

È importante chiudere il client al termine per assicurare che le connessioni WebSocket vengano chiuse dal server e tutte le risorse associate alle connessioni vengano rilasciate. Questo si verifica automaticamente se si chiude il cluster utilizzando `Cluster.close( )`, perché `client.close( )` viene chiamato internamente.

Se il client non viene chiuso correttamente, Neptune termina tutte le connessioni WebSocket inattive dopo 20-25 minuti. Tuttavia, se non chiudi esplicitamente le connessioni WebSocket quando hai finito di usarle e il numero di connessioni live raggiunge il limite di connessioni WebSocket simultanee, le connessioni aggiuntive vengono quindi rifiutate con un codice di errore HTTP 429. A quel punto, è necessario riavviare l'istanza Neptune per chiudere le connessioni.

Il consiglio relativo alla chiamata a `cluster.close()` non si applica alle funzioni AWS Lambda Java. Per informazioni dettagliate, consulta Gestione delle connessioni WebSocket Gremlin nelle funzioni AWS Lambda.

Creazione di una nuova connessione dopo il failover

In caso di failover, Gremlin Driver potrebbe continuare a connettersi al precedente writer perché il nome DNS del cluster viene risolto in un indirizzo IP. In tal caso, puoi creare un nuovo oggetto `Client` dopo il failover.

Impostazione di **`maxInProcessPerConnection`** e **`maxSimultaneousUsagePerConnection`** sullo stesso valore

Entrambi i parametri `maxInProcessPerConnection` e `maxSimultaneousUsagePerConnection` sono correlati al numero massimo di query simultanee che è possibile inviare su una singola connessione WebSocket. Internamente, questi parametri sono correlati e la modifica di uno senza l'altro può portare a un client che riceve un timeout mentre tenta di recuperare una connessione dal pool di connessioni client.

Ti consigliamo di mantenere i valori di utilizzo simultaneo e di elaborazione minimi predefiniti `maxInProcessPerConnection` e di impostare `maxSimultaneousUsagePerConnection` sullo stesso valore.

Il valore su cui impostare questi parametri è una funzione di complessità di query e modello di dati. Un caso d'uso in cui la query restituisce molti dati richiederebbe più larghezza di banda della connessione per query e, nel caso di valori più bassi dei parametri, un valore più elevato per `maxConnectionPoolSize`.

Al contrario, nel caso in cui la query restituisca una quantità di dati inferiore, `maxInProcessPerConnection` e `maxSimultaneousUsagePerConnection` devono essere impostati su un valore più elevato di `maxConnectionPoolSize`.

Invio di query al server come bytecode anziché come stringhe

L'utilizzo di bytecode anziché di una stringa durante l'invio di query comporta dei vantaggi:

- Possibilità di rilevare sintassi di query non valida in anticipo: l'uso della variante bytecode consente di rilevare sintassi di query non valida in fase di compilazione. Se utilizzi la variazione basata su stringa, non individuerai sintassi non valida fino a quando la query non viene inviata al server e viene restituito un errore.
- Possibilità di evitare il rallentamento delle prestazioni basato su stringa: eventuali invii di query basate su stringhe, a prescindere che si utilizzino WebSocket o HTTP, comporta un vertice distaccato, il quale implica che l'oggetto Vertice sia costituito da ID, Etichetta e tutte le proprietà associate con il Vertice (consulta Proprietà di elementi).

  Questo può generare calcoli non necessari nel server nei casi in cui le proprietà non sono richieste. Ad esempio, se il cliente è interessato a ottenere il vertice con l'ID "hakuna#1" utilizzando la query `g.V("hakuna#1")`. Se la query viene inviata come un invio basato su stringa, il server deve recuperare l'ID, l'etichetta e tutte le proprietà per questo vertice. Se la query viene inviata come un invio bytecode, il server deve recuperare l'ID e l'etichetta del vertice.

In altre parole, anziché inviare una query come questa:

```
final Cluster cluster = Cluster.build("localhost")
                               .port(8182)
                               .maxInProcessPerConnection(32)
                               .maxSimultaneousUsagePerConnection(32)
                               .serializer(Serializers.GRAPHBINARY_V1D0)
                               .create();

try {
    final Client client = cluster.connect();
```

```
     List<Result> results =
client.submit("g.V().has('name','pumba').out('friendOf').id()").all().get();
     System.out.println(verticesWithNamePumba);
 } finally {
     cluster.close();
 }
```

Invia le query utilizzando bytecode, come questa:

```
 final Cluster cluster = Cluster.build("localhost")
                                .port(8182)
                                .maxInProcessPerConnection(32)
                                .maxSimultaneousUsagePerConnection(32)
                                .serializer(Serializers.GRAPHBINARY_V1D0)
                                .create();

 try {
     final GraphTraversalSource g =
traversal().withRemote(DriverRemoteConnection.using(cluster));
     List<Object> verticesWithNamePumba = g.V().has("name",
"pumba").out("friendOf").id().toList();
     System.out.println(verticesWithNamePumba);
 } finally {
     cluster.close();
 }
```

Consumare sempre completamente ResultSet o l'iteratore restituito da una query

L'oggetto client deve sempre consumare completamente il `ResultSet` (in caso di invio basato su stringa) o l'iteratore restituito da `GraphTraversal`. Se i risultati della query non sono completamente consumati, il server li mantiene, in attesa che il client finisca di consumarli.

Se la tua applicazione richiede solo una set parziale di risultati, puoi utilizzare una fase `limit(X)` con la query per limitare il numero di risultati generati dal server.

Aggiunta in blocco di vertici e archi in batch

Ogni query inviata al database Neptune viene eseguita nell'ambito di una singola transazione, a meno che non si utilizzi una sessione. Questo significa che se è necessario inserire una notevole

quantità di dati utilizzando le query gremlin, raccogliendole in una dimensione batch di 50-100 è possibile migliorare le prestazioni riducendo il numero di transazioni create per il caricamento.

Ad esempio, aggiungendo 5 vertici al database, l'aspetto è simile al seguente:

```
// Create a GraphTraversalSource for the remote connection
final GraphTraversalSource g =
 traversal().withRemote(DriverRemoteConnection.using(cluster));
// Add 5 vertices in a single query
g.addV("Person").property(T.id, "P1")
 .addV("Person").property(T.id, "P2")
 .addV("Person").property(T.id, "P3")
 .addV("Person").property(T.id, "P4")
 .addV("Person").property(T.id, "P5").iterate();
```

Disabilitazione del caching DNS nella Java Virtual Machine

In un ambiente in cui si desidera bilanciare il carico di richieste su più le repliche di lettura, è necessario disabilitare il caching DNS nella Java Virtual Machine (JVM) e fornire l'endpoint di lettura di Neptune durante la creazione del cluster. La disabilitazione della cache DNS di JVM assicura che il DNS viene risolto nuovamente per ogni nuova connessione, in modo che le richieste vengano distribuite su tutte le repliche di lettura. Puoi eseguire questa operazione nel codice di inizializzazione dell'applicazione con la riga seguente:

```
java.security.Security.setProperty("networkaddress.cache.ttl", "0");
```

Tuttavia, una soluzione più completa e affidabile per il bilanciamento del carico è fornita dal codice client Java Gremlin Amazon in GitHub. Il client Java Gremlin Amazon riconosce la topologia del cluster e distribuisce equamente le connessioni e le richieste tra un set di istanze nel cluster Neptune. Consulta questo post del blog per un esempio di funzione Lambda Java che utilizza tale client.

Facoltativamente, impostare timeout a livello di query

Neptune offre la possibilità di impostare un timeout per le query utilizzando l'opzione del gruppo di parametri `neptune_query_timeout` (consulta Parametri). A partire dalla versione 3.3.7 del client Java, tuttavia, è anche possibile sostituire il timeout globale con codice del tipo seguente:

```
  final Cluster cluster = Cluster.build("localhost")
                              .port(8182)
                              .maxInProcessPerConnection(32)
```

```
                                .maxSimultaneousUsagePerConnection(32)
                                .serializer(Serializers.GRAPHBINARY_V1D0)
                                .create();

 try {
     final GraphTraversalSource g =
traversal().withRemote(DriverRemoteConnection.using(cluster));
     List<Object> verticesWithNamePumba = g.with(ARGS_EVAL_TIMEOUT,
500L).V().has("name", "pumba").out("friendOf").id().toList();
     System.out.println(verticesWithNamePumba);
 } finally {
     cluster.close();
 }
```

Oppure, per invii di query basati su stringa, l'aspetto del codice è simile al seguente:

```
RequestOptions options = RequestOptions.build().timeout(500).create();
List<Result> result = client.submit("g.V()", options).all().get();
```

> **Note**
>
> Potrebbero essere applicati costi imprevisti se viene impostato un valore di timeout delle query troppo elevato, in particolare su un'istanza serverless. Se il timeout non è impostato su un valore ragionevole, la query potrebbe continuare a essere eseguita molto più a lungo del previsto, con conseguenti costi imprevisti. Questo è particolarmente vero per un'istanza serverless, che potrebbe aumentare fino a diventare un tipo di istanza di grandi dimensioni e costosa durante l'esecuzione della query.
>
> Per evitare spese impreviste di questo tipo, specifica un valore di timeout delle query adatto alla maggior parte dei tempi di esecuzione previsti e che causa il timeout solo delle query con tempi di esecuzione insolitamente lunghi.

Soluzione alternativa per un bug keep-alive nei client precedenti alla versione 3.3.4

Solo per versioni client 3.3.3 e precedenti:

Nella versione precedente del client Gremlin esiste un bug che causa l'invio di una nuova richiesta `KeepAlive` al server con ogni query anziché una volta per connessione websocket. Questo

consuma risorse server preziose quando il server elabora richieste keep-alive non necessarie. Per ulteriori informazioni, consulta TINKERPOP-2030.

Per evitare questo problema, eseguire l'aggiornamento al client Java Gremlin versione 3.3.4 o successivo.

La soluzione alternativa è disabilitare il keep-alive dal client impostando il suo intervallo su 0:

```
Cluster.Builder readerBuilder = Cluster.build()
          .port(8182)
          ...
          .keepAliveInterval(0)
```

Risoluzione dei problemi relativi a `java.util.concurrent.TimeoutException`

Il client Java Gremlin genera un'eccezione di tipo `java.util.concurrent.TimeoutException` quando una richiesta Gremlin a livello del client stesso raggiunge il timeout in attesa che uno slot in una delle connessioni WebSocket diventi disponibile. La durata di questo timeout è controllata dal parametro configurabile lato client `maxWaitForConnection`.

> **Note**
>
> Poiché le richieste che raggiungono il timeout a livello del client non vengono mai inviate al server, non si riflettono nelle metriche acquisite dal server, ad esempio `GremlinRequestsPerSec`.

Questo tipo di timeout viene generalmente causato in due modi:

- Il server ha effettivamente raggiunto la capacità massima. In tal caso, la coda nel server si riempie ed è possibile rilevare questo evento monitorando la metrica CloudWatch MainRequestQueuePendingRequests. Il numero di query parallele che il server può gestire dipende dalle dimensioni dell'istanza.

  Se la metrica `MainRequestQueuePendingRequests` non mostra un accumulo di richieste in sospeso nel server, il server può gestire altre richieste e il timeout è causato dalla limitazione delle richieste sul lato client.

- Limitazione delle richieste sul lato client. In genere, questo problema può essere risolto modificando le impostazioni di configurazione del client.

  Il numero massimo di richieste parallele che il client può inviare può essere stimato approssimativamente come segue:

  ```
  maxParallelQueries = maxConnectionPoolSize * Max( maxSimultaneousUsagePerConnection,
   maxInProcessPerConnection )
  ```

  L'invio di un numero di richieste maggiore del valore di `maxParallelQueries` al client causa eccezioni di tipo `java.util.concurrent.TimeoutException`. In genere è possibile risolvere il problema in vari modi:

  - Aumentare la durata del timeout della connessione. Se la latenza non è fondamentale per l'applicazione, aumenta il valore dell'impostazione `maxWaitForConnection` del client. Il client attende quindi più a lungo prima che scada il timeout, aumentando così la latenza.
  - Aumentare il numero massimo di richieste per connessione. In questo modo è possibile inviare un maggior numero di richieste utilizzando la stessa connessione WebSocket. A tale scopo, è necessario aumentare i valori delle impostazioni `maxSimultaneousUsagePerConnection` e `maxInProcessPerConnection` del client. Queste impostazioni devono in genere avere lo stesso valore.
  - Aumentare il numero di connessioni nel pool di connessioni. A tale scopo, è necessario aumentare il valore dell'impostazione `maxConnectionPoolSize` del client. Il costo è costituito da un maggiore consumo di risorse, poiché ogni connessione utilizza la memoria e un descrittore di file del sistema operativo e richiede un handshake SSL e WebSocket durante l'inizializzazione.

Best practice di Neptune per l'utilizzo di openCypher e Bolt

È necessario seguire queste best practice per usare il linguaggio di query openCypher e il protocollo Bolt con Neptune. Per informazioni sull'utilizzo di openCypher in Neptune, consulta Accesso al grafo di Neptune con openCypher.

Preferire archi orientati a quelli bidirezionali nelle query

Quando Neptune esegue ottimizzazioni delle query, gli archi bidirezionali complicano la creazione di piani di query ottimali. I piani non ottimali richiedono al motore di svolgere attività non necessarie riducendo così le prestazioni.

Pertanto, è consigliabile usare archi orientati anziché bidirezionali quando possibile. Usa ad esempio:

```
MATCH p=(:airport {code: 'ANC'})-[:route]->(d) RETURN p)
```

anziché:

```
MATCH p=(:airport {code: 'ANC'})-[:route]-(d) RETURN p)
```

La maggior parte dei modelli di dati non necessita di attraversare gli archi in entrambe le direzioni, quindi è possibile migliorare le prestazioni delle query in modo significativo passando all'utilizzo di archi orientati

Se il tuo modello di dati richiede l'attraversamento di archi bidirezionali, imposta il primo nodo (a sinistra) nel modello `MATCH` come nodo con il filtro più restrittivo.

Prendiamo questo esempio: "Trova tutti gli itinerari (`routes`) da e per l'aeroporto `ANC`". Scrivi questa query per iniziare dall'aeroporto `ANC`, in questo modo:

```
MATCH p=(src:airport {code: 'ANC'})-[:route]-(d) RETURN p
```

Il motore può eseguire la quantità minima di lavoro necessaria per soddisfare la query, perché il nodo con più restrizioni è posizionato come primo nodo (lato sinistro) nel modello e può quindi ottimizzare la query.

Questo approccio è di gran lunga preferibile rispetto all'applicazione del filtro per l'aeroporto `ANC` alla fine del modello, come illustrato di seguito:

```
MATCH p=(d)-[:route]-(src:airport {code: 'ANC'}) RETURN p
```

Quando il nodo con più restrizioni non viene posizionato per primo nel modello, il motore deve eseguire attività aggiuntive perché non può ottimizzare la query e deve eseguire ulteriori ricerche per generare i risultati.

Neptune non supporta più query simultanee in una transazione

Sebbene il driver Bolt stesso consenta query simultanee in una transazione, Neptune non supporta l'esecuzione simultanea di più query in una transazione. Neptune richiede invece che più query in una transazione vengano eseguite in sequenza e che i risultati di ciascuna query vengano utilizzati completamente prima che venga avviata la query successiva.

L'esempio seguente mostra come usare Bolt per eseguire più query in sequenza in una transazione, affinché i risultati di ognuna di esse vengano utilizzati completamente prima dell'inizio di quella successiva:

```
final String query = "MATCH (n) RETURN n";

try (Driver driver = getDriver(HOST_BOLT, getDefaultConfig())) {
  try (Session session = driver.session(readSessionConfig)) {
    try (Transaction trx = session.beginTransaction()) {
      final Result res_1 = trx.run(query);
      Assert.assertEquals(10000, res_1.list().size());
      final Result res_2 = trx.run(query);
      Assert.assertEquals(10000, res_2.list().size());
    }
  }
}
```

Creazione di una nuova connessione dopo il failover

In caso di failover, il driver Bolt può continuare a connettersi all'istanza di scrittura precedente anziché alla nuova istanza attiva, poiché il nome DNS è stato risolto in un indirizzo IP specifico.

Per evitare questo scenario, chiudi e riconnetti l'oggetto `Driver` dopo ogni failover.

Gestione delle connessioni per applicazioni di lunga durata

Quando crei applicazioni di lunga durata, come quelle in esecuzione all'interno di container o su istanze Amazon EC2, crea un'istanza di un oggetto `Driver` una volta e poi riutilizza tale oggetto per tutta la durata dell'applicazione. L'oggetto `Driver` è thread-safe e l'overhead di inizializzazione è notevole.

Gestione delle connessioni per AWS Lambda

L'uso dei driver Bolt non è consigliato all'interno delle funzioni AWS Lambda, a causa del sovraccarico delle connessioni e dei requisiti di gestione. Usa invece l'endpoint HTTPS.

Chiusura degli oggetti driver al termine dell'utilizzo

È importante chiudere il client al termine dell'utilizzo per assicurarsi che le connessioni Bolt vengano chiuse dal server e che tutte le risorse associate alle connessioni vengano rilasciate. Ciò avviene automaticamente se si chiude il driver utilizzando `driver.close()`.

Se il driver non viene chiuso correttamente, Neptune interrompe tutte le connessioni Bolt inattive dopo 20 minuti o dopo 10 giorni se si utilizza l'autenticazione IAM.

Neptune non supporta più di 1.000 connessioni Bolt simultanee. Se non chiudi esplicitamente le connessioni quando hai finito di usarle e il numero di connessioni live raggiunge il limite di 1.000, qualsiasi nuovo tentativo di connessione avrà esito negativo.

Utilizzo di modalità di transazione esplicite per la lettura e la scrittura

Quando si utilizzano transazioni con Neptune e il driver Bolt, è preferibile impostare esplicitamente la modalità di accesso per le transazioni di lettura e scrittura sui valori corretti.

Transazioni di sola lettura

Per le transazioni di sola lettura, se non si passa la configurazione della modalità di accesso appropriata durante la creazione della sessione, viene utilizzato il livello di isolamento predefinito, ovvero l'isolamento delle query di mutazione. Di conseguenza, è importante impostare la modalità di accesso a `read` in modo esplicito per le transazioni di sola lettura.

Esempio di transazione di lettura con commit automatico:

```
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.READ)
  .build();
Session session = driver.session(sessionConfig);
try {
  (Add your application code here)
} catch (final Exception e) {
  throw e;
} finally {
  driver.close()
}
```

Esempio di transazione di lettura:

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withDefaultAccessMode(AccessMode.READ)
  .build();
```

```
driver.session(sessionConfig).readTransaction(
  new TransactionWork<List<String>>() {
    @Override
    public List<String> execute(org.neo4j.driver.Transaction tx) {
      (Add your application code here)
    }
  }
);
```

In entrambi i casi, l'isolamento SNAPSHOT viene ottenuto utilizzando la semantica delle transazioni di sola lettura di Neptune.

Poiché le repliche di lettura accettano solo query di sola lettura, qualsiasi query inviata a una replica di lettura viene eseguita in base alla semantica di isolamento SNAPSHOT.

Non esistono letture dirty o letture non ripetibili per le transazioni di sola lettura.

Transazioni di sola lettura

Per le query di mutazione, esistono tre diversi meccanismi per creare una transazione di scrittura, ognuno dei quali è illustrato di seguito:

Esempio di transazione di scrittura implicita:

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();
driver.session(sessionConfig).writeTransaction(
  new TransactionWork<List<String>>() {
    @Override
    public List<String> execute(org.neo4j.driver.Transaction tx) {
      (Add your application code here)
    }
  }
);
```

Esempio di transazione di scrittura con commit automatico:

```
SessionConfig sessionConfig = SessionConfig
  .builder()
```

```
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.Write)
  .build();
Session session = driver.session(sessionConfig);
try {
  (Add your application code here)
} catch (final Exception e) {
    throw e;
} finally {
    driver.close()
}
```

Esempio di transazione di scrittura esplicita:

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();
Transaction beginWriteTransaction = driver.session(sessionConfig).beginTransaction();
  (Add your application code here)
beginWriteTransaction.commit();
driver.close();
```

Livelli di isolamento per le transazioni di scrittura

- Le letture effettuate come parte delle query di mutazione vengono eseguite nell'isolamento delle transazioni `READ COMMITTED`.
- Non esistono letture dirty per le letture effettuate come parte delle query di mutazione.
- I record e gli intervalli di record vengono bloccati durante la lettura in una query di mutazione.
- Quando un intervallo dell'indice è stato letto da una transazione di mutazione, esiste una forte garanzia che questo intervallo non verrà modificato da alcuna transazione simultanea fino al termine della lettura.

Le query di mutazione non sono thread-safe.

Per i conflitti, consulta Risoluzione dei conflitti tramite timeout di attesa di blocco.

Per le query di mutazione i tentativi non vengono ripetuti automaticamente in caso di errore.

Logica di ripetizione dei tentativi per le eccezioni

Per tutte le eccezioni che consentono un nuovo tentativo, in genere è preferibile utilizzare una strategia di ripetizione dei tentativi e backoff esponenziale che preveda tempi di attesa progressivamente più lunghi tra un tentativo e l'altro, in modo da gestire al meglio i problemi temporanei, come gli errori `ConcurrentModificationException`. Di seguito viene illustrato un esempio di modello di ripetizione dei tentativi e backoff esponenziale:

```
public static void main() {
  try (Driver driver = getDriver(HOST_BOLT, getDefaultConfig())) {
    retriableOperation(driver, "CREATE (n {prop:'1'})")
        .withRetries(5)
        .withExponentialBackoff(true)
        .maxWaitTimeInMilliSec(500)
        .call();
  }
}

protected RetryableWrapper retriableOperation(final Driver driver, final String query){
  return new RetryableWrapper<Void>() {
    @Override
    public Void submit() {
      log.info("Performing graph Operation in a retry manner......");
      try (Session session = driver.session(writeSessionConfig)) {
        try (Transaction trx =  session.beginTransaction()) {
            trx.run(query).consume();
            trx.commit();
        }
      }
      return null;
    }

    @Override
    public boolean isRetryable(Exception e) {
      if (isCME(e)) {
        log.debug("Retrying on exception.... {}", e);
        return true;
      }
      return false;
    }

    private boolean isCME(Exception ex) {
```

```
      return ex.getMessage().contains("Operation failed due to conflicting concurrent
 operations");
    }
  };
}



/**
 * Wrapper which can retry on certain condition. Client can retry operation using this
 class.
 */
@Log4j2
@Getter
public abstract class RetryableWrapper<T> {

  private long retries = 5;
  private long maxWaitTimeInSec = 1;
  private boolean exponentialBackoff = true;

  /**
   * Override the method with custom implementation, which will be called in retryable
 block.
   */
  public abstract T submit() throws Exception;

  /**
   * Override with custom logic, on which exception to retry with.
   */
  public abstract boolean isRetryable(final Exception e);

  /**
   * Define the number of retries.
   *
   * @param retries -no of retries.
   */
  public RetryableWrapper<T> withRetries(final long retries) {
    this.retries = retries;
    return this;
  }

  /**
   * Max wait time before making the next call.
   *
```

```
   * @param time - max polling interval.
   */
  public RetryableWrapper<T> maxWaitTimeInMilliSec(final long time) {
    this.maxWaitTimeInSec = time;
    return this;
  }

  /**
   * ExponentialBackoff coefficient.
   */
  public RetryableWrapper<T> withExponentialBackoff(final boolean expo) {
    this.exponentialBackoff = expo;
    return this;
  }

  /**
   * Call client method which is wrapped in submit method.
   */
  public T call() throws Exception {
    int count = 0;
    Exception exceptionForMitigationPurpose = null;
    do {
      final long waitTime = exponentialBackoff ? Math.min(getWaitTimeExp(retries),
maxWaitTimeInSec) : 0;
      try {
          return submit();
      } catch (Exception e) {
        exceptionForMitigationPurpose = e;
        if (isRetryable(e) && count < retries) {
          Thread.sleep(waitTime);
          log.debug("Retrying on exception attempt - {} on exception cause - {}",
count, e.getMessage());
        } else if (!isRetryable(e)) {
          log.error(e.getMessage());
          throw new RuntimeException(e);
        }
      }
    } while (++count < retries);

    throw new IOException(String.format(
          "Retry was unsuccessful.... attempts %d. Hence throwing exception " + "back
to the caller...", count),
          exceptionForMitigationPurpose);
  }
```

```
  /*
   * Returns the next wait interval, in milliseconds, using an exponential backoff
   * algorithm.
   */
  private long getWaitTimeExp(final long retryCount) {
    if (0 == retryCount) {
      return 0;
    }
    return ((long) Math.pow(2, retryCount) * 100L);
  }
}
```

Best practice di Neptune per l'utilizzo di SPARQL

Segui queste best practice durante l'utilizzo del linguaggio di query SPARQL con Neptune. Per ulteriori informazioni sull'utilizzo di SPARQL in Neptune, consulta Accesso al grafo Neptune con SPARQL.

Query di tutti i grafi nominati per impostazione predefinita

Amazon Neptune associa ogni tripla a un grafo denominato. Il grafo predefinito è definito come l'unione di grafi denominati.

Se invii una query SPARQL senza specificare un grafo in maniera esplicita tramite la parola chiave `GRAPH` o costrutti quali `FROM NAMED`, Neptune considera sempre tutte le triple nell'istanza database. Ad esempio, la seguente query restituisce tutte le triple da un endpoint SPARQL Neptune:

```
SELECT * WHERE { ?s ?p ?o }
```

Le triple che appaiono in più grafi vengono restituite una sola volta.

Per informazioni sulla specifica del grafico predefinito, consulta la sezione RDF Dataset delle specifiche SPARQL 1.1 Query Language.

Specifica di un grafo denominato per il caricamento

Amazon Neptune associa ogni tripla a un grafo denominato. Se non specifichi un grafo denominato durante il caricamento, l'inserimento o l'aggiornamento di triple, Neptune utilizza il grafo

denominato di fallback definito dall'URI, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Se usi lo strumento di caricamento in blocco Neptune, puoi specificare il grafo denominato da utilizzare per tutte le triple (o quadruple con la quarta posizione vuota) con il parametro `parserConfiguration: namedGraphUri`. Per ulteriori informazioni sulla sintassi del comando `Load` dello strumento di caricamento in blocco Neptune, consulta the section called “Comando dello strumento di caricamento”.

Scelta tra FILTER, FILTER...IN e VALUES nelle query

Sono disponibili tre tecniche di base per inserire valori in query SPARQL: `FILTER`, `FILTER...IN` e `VALUES`.

Ad esempio, potresti voler cercare gli amici di più persone con una singola query. Utilizzando `FILTER`, è possibile strutturare la query come segue:

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
WHERE {?s foaf:knows ?o. FILTER (?s = ex:person1 || ?s = ex:person2)}
```

Questo restituisce nel grafo tutte le triple che hanno `?s` associato a `ex:person1` o `ex:person2` e hanno un arco in uscita etichettato come `foaf:knows`.

Puoi anche creare una query utilizzando `FILTER...IN` che restituisce risultati equivalenti:

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
WHERE {?s foaf:knows ?o. FILTER (?s IN (ex:person1, ex:person2))}
```

Puoi anche creare una query utilizzando `VALUES`, che in questo caso restituisce risultati equivalenti:

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
```

```
  WHERE {?s foaf:knows ?o. VALUES ?s {ex:person1 ex:person2}}
```

Anche se in molti casi queste query sono semanticamente equivalenti, vi sono alcuni casi in cui le due varianti `FILTER` differiscono dalla variante `VALUES`:

- Il primo caso corrisponde a quando inserisci valori duplicati, come ad esempio inserire la stessa persona due volte. In questo caso, la query `VALUES` include il duplicato nel risultato. Puoi eliminare esplicitamente questi duplicati aggiungendo `DISTINCT` alla clausola `SELECT`. In alcuni casi, potresti volere i duplicati nei risultati della query per inserire valori ridondanti.

  Tuttavia, quando lo stesso valore appare più volte, le versioni `FILTER` e `FILTER...IN` lo estraggono una sola volta.
- Il secondo caso è correlato al fatto che `VALUES` esegue sempre una corrispondenza esatta, mentre in alcuni casi `FILTER` potrebbe applicare la promozione di tipo ed eseguire corrispondenze parziali.

  Ad esempio, quando includi un valore letterale come `"2.0"^^xsd:float` nella clausola dei valori, una query `VALUES` esegue la corrispondenza esatta con questo valore letterale, inclusi il valore letterale e il tipo di dati.

  Al contrario, `FILTER` produce una corrispondenza parziale per questi valori letterali numerici. Le corrispondenze possono includere valori letterali con lo stesso valore ma con diversi tipi di dati numerici, come `xsd:double`.

  > **Note**
  >
  > Non vi è alcuna differenza tra il comportamento di `FILTER` e `VALUES` durante l'enumerazione di valori letterali di stringa o URI.

Le differenze tra `FILTER` e `VALUES` possono influenzare l'ottimizzazione e la strategia di valutazione query risultante. A meno che il caso d'uso richieda le corrispondenze parziali, consigliamo di utilizzare `VALUES` poiché evita di controllare i casi speciali relativi al casting del tipo. Di conseguenza, `VALUES` spesso produce una query più efficiente, più veloce e meno costosa.

Limiti per Amazon Neptune

Regioni

Amazon Neptune è disponibile nelle seguenti regioni: AWS

- Stati Uniti orientali (Virginia settentrionale): `us-east-1`
- Stati Uniti orientali (Ohio): `us-east-2`
- Stati Uniti occidentali (California settentrionale): `us-west-1`
- Stati Uniti occidentali (Oregon): `us-west-2`
- Canada (Centrale): `ca-central-1`
- Sud America (San Paolo): `sa-east-1`
- Europa (Stoccolma): `eu-north-1`
- Europa (Irlanda): `eu-west-1`
- Europa (Londra): `eu-west-2`
- Europa (Parigi): `eu-west-3`
- Europa (Francoforte): `eu-central-1`
- Medio Oriente (Bahrein): `me-south-1`
- Medio Oriente (Emirati Arabi Uniti): `me-central-1`
- Israele (Tel Aviv): `il-central-1`
- Africa (Città del Capo): `af-south-1`
- Asia Pacifico (Hong Kong): `ap-east-1`
- Asia Pacifico (Tokyo): `ap-northeast-1`
- Asia Pacifico (Seoul): `ap-northeast-2`
- Asia Pacifico (Osaka): `ap-northeast-3`
- Asia Pacifico (Singapore): `ap-southeast-1`
- Asia Pacifico (Sydney): `ap-southeast-2`
- Asia Pacifico (Mumbai): `ap-south-1`
- Cina (Pechino): `cn-north-1`
- Cina (Ningxia): `cn-northwest-1`
- AWS GovCloud (Stati Uniti occidentali): `us-gov-west-1`

- AWS GovCloud (Stati Uniti orientali): `us-gov-east-1`

Differenze nelle regioni della Cina

Come per molti AWS servizi, Amazon Neptune opera in modo leggermente diverso in Cina (Pechino) e Cina (Ningxia) rispetto ad altre regioni. AWS

Ad esempio, quando Neptune ML utilizza Gateway Amazon API per creare il proprio servizio di esportazione, l'autenticazione IAM è abilitata per impostazione predefinita. Nelle regioni della Cina, il processo di modifica di tale opzione è leggermente diverso rispetto ad altre regioni.

Queste e altre differenze sono illustrate qui.

Dimensione massima dei volumi del cluster di archiviazione

Un volume di cluster Neptune può crescere fino a una dimensione massima di 128 tebibyte (TiB) in tutte le regioni supportate tranne la Cina e GovCloud, dove il limite è di 64 TiB. Questo vale per tutti i rilasci del motore a partire da Rilascio: 1.0.2.2 (09/03/2020). Per informazioni, consulta Archiviazione, affidabilità e disponibilità di Amazon Neptune.

Dimensioni dell'istanza database supportate

Neptune supporta diverse classi di istanze DB in diverse regioni. AWS Per scoprire quali classi sono supportate in una determinata regione, consulta Prezzi di Amazon Neptune e scegli la regione di interesse.

Limiti per ogni account AWS

Per alcune funzionalità di gestione, Amazon Neptune utilizza la tecnologia operativa condivisa con Amazon Relational Database Service (Amazon RDS).

Ogni AWS account ha dei limiti per ogni regione sul numero di risorse Amazon Neptune e Amazon RDS che puoi creare. Queste risorse includono le istanze database e i cluster di database.

Dopo aver raggiunto il limite per una risorsa, le ulteriori richieste di creazione di tale risorsa restituiranno un errore con un'eccezione.

Per un elenco dei limiti condivisi tra Amazon Neptune e Amazon RDS, consulta Limiti di Amazon RDS nella Guida per l'utente di Amazon.

La connessione a Neptune richiede un VPC

Amazon Neptune è un servizio esclusivo del cloud privato virtuale (VPC).

Inoltre, le istanze non consentono l'accesso dall'esterno del VPC.

Neptune richiede SSL

A partire dalla versione `1.0.4.0` del motore, Amazon Neptune consente solo connessioni Secure Sockets Layer (SSL) tramite HTTPS a qualsiasi istanza o endpoint del cluster.

Neptune richiede TLS versione 1.2, che utilizza le seguenti suite di crittografia avanzate:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

Zone di disponibilità e gruppi di sottoreti del database

Amazon Neptune richiede un gruppo di sottoreti del database per ogni cluster che ha sottoreti in almeno due zone di disponibilità (AZ) supportate.

Ti consigliamo di usare tre o più sottoreti in diverse zone di disponibilità.

Massimo payload delle richieste HTTP (150 MB)

La dimensione totale delle richieste HTTP Gremlin e SPARQL deve essere inferiore a 150 MB. Se una richiesta supera questa dimensione, Neptune restituisce `HTTP 400: BadRequestException`.

Questo limite non si applica alle connessioni Gremlin. WebSockets

Differenze di implementazione di Gremlin

L'implementazione di Gremlin in Amazon Neptune presenta specifici dettagli di implementazione che possono essere diversi da altre implementazioni Gremlin.

Per ulteriori informazioni, consulta Conformità agli standard Gremlin in Amazon Neptune.

Neptune non supporta caratteri null nei dati di stringa

Neptune non supporta caratteri null nei dati di stringa. Questo vale per i dati del grafo delle proprietà per Gremlin e openCypher e per i dati RDF/SPARQL.

SPARQL UPDATE LOAD da URI

`SPARQL UPDATE LOAD` da URI funziona solo con le risorse all'interno dello stesso VPC.

Sono inclusi gli URL Amazon S3 nella stessa regione del cluster per cui è stato creato un endpoint VPC Amazon S3.

L'URL Amazon S3 deve essere HTTPS e qualsiasi autenticazione deve essere inclusa nell'URL. Per ulteriori informazioni, consulta Autenticazione delle richieste: Utilizzo dei parametri di query nella Documentazione di riferimento delle API di Amazon Simple Storage Service.

Per informazioni su come creare un endpoint VPC, vedi Creazione di un endpoint VPC Amazon S3.

Se hai bisogno di caricare dati da un file, è consigliabile utilizzare l'API dello strumento di caricamento di Neptune. Per ulteriori informazioni, consulta Uso dello strumento di caricamento in blocco Amazon Neptune per importare i dati.

> **Note**
>
> L'API dello strumento di caricamento di Amazon Neptune è non ACID.

Autenticazione e controllo degli accessi IAM

Nelle versioni del motore Neptune precedenti al rilascio 1.2.0.0, l'autenticazione e il controllo degli accessi IAM sono supportati solo a livello di cluster database. Dal rilascio `1.2.0.0` in poi, tuttavia, è possibile controllare l'accesso basato su query a un livello più granulare utilizzando le chiavi di condizione nelle policy IAM. Per ulteriori informazioni, consulta Utilizzo delle azioni di query nelle dichiarazioni di policy di accesso ai dati di Neptune e Panoramica di AWS Identity and Access Management (IAM) in Amazon Neptune.

La console Amazon Neptune richiede autorizzazioni. NeptuneReadOnlyAccess Puoi limitare l'accesso agli utenti IAM revocando questo accesso. Per ulteriori informazioni, consultare AWS politiche gestite (predefinite) per Amazon Neptune

Amazon Neptune non supporta il controllo degli accessi basato su nome utente/password.

WebSocket connessioni simultanee e tempo massimo di connessione

Esiste un limite al numero di WebSocket connessioni simultanee per istanza DB di Neptune. Quando viene raggiunto tale limite, Neptune limita qualsiasi richiesta di apertura di una WebSocket nuova connessione per evitare di utilizzare tutta la memoria heap allocata.

Per tutti i tipi di istanze più grandi supportati da Neptune e tutte le istanze serverless, il numero massimo di connessioni simultanee WebSocket è 32.000 (32.768).

Il numero massimo di WebSocket connessioni simultanee per i tipi di istanze più piccoli è elencato nella tabella seguente:

Tipo di istanza	Numero massimo di connessioni simultanee WebSocket
db.t3.medium	512
db.t4g.medium	512
db.r5.large	2.048
db.r5d.large	2.048
db.r5.xlarge	4,096
db.r5.2xlarge	8,192
db.r5d.2xlarge	8,192
db.r5.4xlarge	16,384
db.r5d.4xlarge	16,384

Tipo di istanza	Numero massimo di connessioni simultanee WebSocket
db.r6g.large	2.048
db.r6gd.large	2.048
db.r6g.xlarge	4,096
db.r6gd.xlarge	4,096
db.r6g.2xlarge	8,192
db.r6gd.2xlarge	8,192
db.r6g.4xlarge	16,384
db.r6gd.4xlarge	16,384
db.x2g.large	2.048
db.x2gd.large	2.048
db.x2g.xlarge	4,096
db.x2gd.xlarge	4,096
db.x2iedn.xlarge	4,096
db.x2g.2xlarge	8,192
db.x2gd.2xlarge	8,192
db.x2g.4xlarge	16,384
db.x2gd.4xlarge	16,384
db.x2iedn.2xlarge	16,384
db.x2iezn.2xlarge	16,384
serverless	32,768

Tipo di istanza	Numero massimo di connessioni simultanee WebSocket
(altri tipi di istanza di grandi dimensioni)	32,768

> **Note**
>
> A partire dal rilascio 1.1.0.0 del motore Neptune, i tipi di istanza R4 non sono più supportati.

Quando un client chiude correttamente una connessione, la chiusura si riflette immediatamente nel conteggio delle connessioni aperte.

Se il client non chiude una connessione, la connessione può essere chiusa automaticamente dopo un timeout di inattività da 20 a 25 minuti (il timeout di inattività è il tempo trascorso dalla ricezione dell'ultimo messaggio dal client). Tuttavia, finché non viene raggiunto il timeout di inattività, Neptune mantiene la connessione aperta a tempo indeterminato.

Quando l'autenticazione IAM è abilitata, una WebSocket connessione viene sempre disconnessa alcuni minuti e più di 10 giorni dopo essere stata stabilita, se non è già stata chiusa entro tale data.

Limiti per proprietà ed etichette

Non c'è limite al numero di vertici e bordi o quad RDF che puoi avere in un grafico.

Non esiste inoltre alcun limite al numero di proprietà o etichette che un vertice o un bordo può avere.

Esiste un limite di dimensioni pari a 55 MB sulla dimensione di una singola proprietà o etichetta. In termini di RDF, ciò significa che il valore in qualsiasi colonna (S, P, O o G) di un quad RDF non può superare i 55 MB.

Se è necessario associare un oggetto più grande come un'immagine a un vertice o un nodo nel grafo, è possibile archiviarlo come file in Amazon S3 e utilizzare il percorso di Amazon S3 come proprietà o etichetta.

Limiti che interessano lo strumento di caricamento in blocco Neptune

Non è possibile accodare più di 64 processi di caricamento in blocco alla volta.

Neptune tiene traccia solo dei 1.024 processi di caricamento in blocco più recenti.

Neptune archivia solo gli ultimi 10.000 dettagli di errore per processo.

Utilizzo di altri servizi AWS

Puoi usare Amazon Neptune in combinazione con molti altri servizi AWS.

Integrazioni di Neptune con altri servizi

- AWS Glue: AWS Glue è un servizio di integrazione dei dati serverless che consente di eseguire processi di estrazione, trasformazione e caricamento (ETL) sui dati.

  Neptune fornisce una libreria open source, neptune-python-utilities, che semplifica l'uso di Python e Gremlin all'interno di un processo Glue. Il connettore Neo4j per Spark è supportato anche per l'esecuzione di processi Scala e openCypher Glue.
- Amazon SageMaker: Amazon SageMaker è una piattaforma di machine learning completa per la creazione, l'addestramento e l'implementazione di modelli di machine learning di alta qualità.

  Neptune si integra con SageMaker in due modi principali:

  - Neptune fornisce un pacchetto Python open source per i notebook Jupyter disponibile nel progetto notebook per i grafi Neptune in GitHub. Questo pacchetto contiene un set di comandi magic di Jupyter, notebook con tutorial ed esempi di codice che forniscono un ambiente di sviluppo del codice interattivo in cui è possibile apprendere i concetti sulla tecnologia dei grafi e su Neptune. Neptune fornisce un ambiente completamente gestito per i notebook Jupyter ospitati da SageMaker e si collega automaticamente ai notebook nel progetto notebook per i grafi Neptune open source.
  - La funzionalità Neptune ML consente di creare e addestrare modelli di machine learning utili su grafi di grandi dimensioni in poche ore anziché in settimane. A tale scopo, Neptune ML utilizza la tecnologia delle reti neurali a grafo (GNN) basata su Amazon SageMaker e Deep Graph Library (DGL).
- AWS Lambda: le funzioni AWS Lambda hanno molti utilizzi nelle applicazioni Neptune.

  Per informazioni su come utilizzare le funzioni Lambda con i driver e le varianti del linguaggio Gremlin più diffusi e per esempi specifici di funzioni Lambda scritte in Java, JavaScript e Python, consulta Utilizzo delle funzioni AWS Lambda in Amazon Neptune.
- Amazon Athena: Amazon Athena è un servizio di query interattivo che semplifica l'analisi di dati direttamente in Amazon Simple Storage Service (Amazon S3) e altre origini dati federate con SQL standard.

Neptune fornisce un connettore per Athena che consente ad Athena di comunicare con i dati archiviati in Neptune.

- AWS Database Migration Service (AWS DMS): AWS Database Migration Service è un servizio web AWS che è possibile utilizzare per eseguire la migrazione dei dati da un database all'altro.

  AWS DMS può caricare i dati in Neptune dai database di origine supportati in modo rapido e sicuro. Il database di origine resterà completamente operativo anche durante la migrazione, per ridurre al minimo le interruzioni delle applicazioni che lo utilizzano.

- AWS Backup: AWS Backup è un servizio di backup completamente gestito che semplifica la centralizzazione e l'automazione del backup dei dati in tutti i servizi AWS nel cloud e on-premise.

  AWS Backup consente di creare snapshot automatici periodici dei cluster Neptune utilizzando la policy di protezione dei dati centralizzata tra i servizi AWS supportati per database, archiviazione e calcolo.

- AWS SDK per pandas: AWS SDK per pandas (precedentemente noto come AWS Data Wrangler o `awswrangler`), è un'iniziativa Python open source di AWS Professional Services che estende la potenza della libreria di analisi dei dati Python `pandas` a AWS, connettendo `DataFrames` e più di 30 servizi AWS correlati ai dati, incluso Neptune.

  Oltre all'SDK, sono disponibili anche un tutorial su come utilizzarlo con Neptune e diversi notebook Neptune di esempio, tra cui Fraud Ring Detection, Synthetic Identity Detection e Logistics Analysis.

- Driver JDBC: il driver JDBC Neptune supporta le query openCypher, Gremlin, SQL-Gremlin e SPARQL.

  La connettività JDBC semplifica la connessione a Neptune con strumenti di business intelligence (BI) come Tableau.

Strumenti e utilità per Neptune

Amazon Neptune offre una serie di strumenti e utilità in grado di semplificare e automatizzare il lavoro con un grafo. Di seguito sono riportati alcuni esempi:

Strumenti Amazon Neptune

- Utilità Amazon Neptune per GraphQL: l'utilità Amazon Neptune per GraphQL è uno strumento a riga di comando Node.js open source che può aiutarti a creare e gestire un'API GraphQL per un database a grafo delle proprietà di Neptune. È un modo per creare, senza codice, un resolver GraphQL per query GraphQL che hanno un numero variabile di parametri di input e restituiscono un numero variabile di campi annidati.

Utilità Amazon Neptune per GraphQL

L'utilità Amazon Neptune per GraphQL è uno strumento a riga di comando Node.js open source che può aiutarti a creare e gestire un'API GraphQL per un database a grafo delle proprietà di Neptune (non funziona ancora con dati RDF). È un modo per creare, senza codice, un resolver GraphQL per query GraphQL che hanno un numero variabile di parametri di input e restituiscono un numero variabile di campi annidati.

È stato rilasciato come progetto open source all'indirizzo https://github.com/aws/ amazon-neptune-for-graphql.

Puoi installare l'utilità usando NPM in questo modo (consulta la sezione Installazione e configurazione per i dettagli):

```
npm i @aws/neptune-for-graphql -g
```

L'utilità può scoprire lo schema di un grafo delle proprietà di Neptune esistente, inclusi nodi, archi, proprietà e cardinalità degli archi. Quindi genera uno schema GraphQL con le direttive necessarie per mappare i tipi GraphQL ai nodi e agli archi del database, poi genera automaticamente il codice resolver. Quest'ultimo è progettato per ridurre al minimo la latenza restituendo solo i dati richiesti dalla query GraphQL.

Puoi anche iniziare con uno schema GraphQL esistente e un database Neptune vuoto, lasciando che l'utilità deduca le direttive necessarie per mappare lo schema GraphQL ai nodi e agli archi dei dati da

caricare nel database. In alternativa, puoi iniziare con uno schema GraphQL e le direttive che hai già creato o modificato.

L'utilità è in grado di creare tutte le AWS risorse necessarie per la sua pipeline, tra cui l'AWS AppSync API, i ruoli IAM, l'origine dati, lo schema e il resolver e la AWS funzione Lambda che interroga Neptune.

> **Note**
>
> Gli esempi da riga di comando riportati qui presuppongono l'uso di una console Linux. Se usi Windows, sostituisci la barra rovesciata ("\") alla fine delle righe con un accento circonflesso ("^").

Argomenti

Installazione e configurazione dell'utilità Amazon Neptune per GraphQL

Se intendi usare l'utilità con un database Neptune esistente, ne hai bisogno per poterti connettere all'endpoint del database. Per impostazione predefinita, un database Neptune è accessibile solo dall'interno del VPC in cui è contenuto.

Poiché l'utilità è uno strumento da riga di comando Node.js, per poterla eseguire è necessario che sia installato Node.js (versione 18 o successiva). Per installare Node.js su un'istanza EC2 nello stesso VPC che contiene il database Neptune, segui le istruzioni riportate qui. L'istanza di dimensione minima per eseguire l'utilità è t2.micro. Durante la creazione dell'istanza, seleziona il VPC del database Neptune dal menu a discesa Gruppi di sicurezza comuni.

Tuttavia, a partire dalla versione del motore 1.2.0.0, puoi creare un endpoint pubblico per il database Neptune accessibile all'esterno del VPC. Se hai creato un endpoint pubblico, puoi installare Node.js e l'utilità sul tuo computer locale. Per installare Node.js su macOS o Windows, visita il sito Web di Node.js per scaricare il programma di installazione.

Per installare l'utilità stessa su un'istanza EC2 o sul computer locale, usa NPM:

```
npm install aws-neptune-for-graphql -g
```

Puoi quindi eseguire il comando help dell'utilità per verificare se è stata installata correttamente:

```
neptune-for-graphql --help
```

Puoi anche decidere di installare l'interfaccia AWS CLI per gestire le risorse AWS.

Scansione dei dati in un database Neptune esistente

Che tu abbia o meno familiarità con GraphQL, il comando seguente è il modo più veloce per creare un'API GraphQL. Questa procedura presuppone che sia stata installata e configurata l'utilità Neptune per GraphQL come descritto nella sezione dedicata all'installazione, in modo che sia connessa all'endpoint del database Neptune.

```
neptune-for-graphql \
  --input-graphdb-schema-neptune-endpoint (your neptune database endpoint):(port
 number) \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (your new GraphQL API name) \
  --output-resolver-query-https
```

L'utilità analizza il database per scoprire lo schema dei nodi, degli archi e delle proprietà in esso contenuti. Sulla base di tale schema, deduce uno schema GraphQL con le query e le mutazioni associate. Quindi crea un'API AppSync GraphQL e le AWS risorse necessarie per utilizzarla. Queste risorse includono un paio di ruoli IAM e una funzione Lambda contenente il codice resolver GraphQL.

Al termine dell'utilità, troverai una nuova API GraphQL nella AppSync console con il nome assegnato nel comando. Per testarlo, usa l'opzione AppSync Queries del menu.

Se esegui nuovamente lo stesso comando dopo aver aggiunto altri dati al database, aggiornerà l' AppSync API e il codice Lambda di conseguenza.

Per rilasciare tutte le risorse associate al comando, esegui:

```
neptune-for-graphql \
  --remove-aws-pipeline-name (your new GraphQL API name from above)
```

Inizio da uno schema GraphQL senza direttive

Puoi iniziare da un database Neptune vuoto e usare uno schema GraphQL senza direttive per creare i dati e interrogarli. Il seguente comando crea automaticamente le risorse AWS per eseguire questo processo:

```
neptune-for-graphql \
  --input-schema-file (your GraphQL schema file)
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (name for your new GraphQL API) \
  --create-update-aws-pipeline-neptune-endpoint (your Neptune database endpoint):(port
 number) \
  --output-resolver-query-https
```

Il file dello schema GraphQL deve includere i tipi di schema GraphQL, come mostrato nell'esempio per TODO riportato di seguito. L'utilità analizza lo schema e crea una versione estesa basata sui tipi che usi. Aggiunge query e mutazioni per i nodi archiviati nel database a grafo e, se lo schema ha dei tipi annidati, aggiunge relazioni tra i tipi archiviati come archi nel database.

L'utilità crea un'API AppSync GraphQL e tutte le AWS risorse richieste. Queste includono un paio di ruoli IAM e una funzione Lambda che contiene il codice resolver GraphQL. Al termine del comando, puoi trovare una nuova API GraphQL con il nome specificato nella AppSync console. Per testarlo, usa Queries nel menu. AppSync

Nell'esempio seguente viene illustrato il funzionamento:

Esempio Todo, inizio da uno schema GraphQL senza direttive

In questo esempio, partiamo da uno schema GraphQL di tipo Todo senza direttive, disponibile nella directory *???samples???*. Sono inclusi questi due tipi:

```
type Todo {
  name: String
  description: String
  priority: Int
  status: String
  comments: [Comment]
}

type Comment {
  content: String
```

```
}
```

Questo comando elabora lo schema Todo e un endpoint di un database Neptune vuoto per creare un'API GraphQL in: AWS AppSync

```
neptune-for-graphql /
  --input-schema-file ./samples/todo.schema.graphql \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name TodoExample \
  --create-update-aws-pipeline-neptune-endpoint (empty Neptune database endpoint):(port
 number) \
  --output-resolver-query-https
```

L'utilità crea un nuovo file nella cartella di output denominata `TodoExample.source.graphql` e l'API GraphQL in. AppSync L'utilità deduce quanto segue:

- Nel tipo Todo è stato aggiunto `@relationship` un nuovo CommentEdge tipo. Questo indica al resolver di connettere Todo a Comment usando un database grafico chiamato edge. CommentEdge
- Ha aggiunto un nuovo input chiamato TodoInput per aiutare le domande e le mutazioni.
- Aggiunta di due query per ogni tipo (Todo, Comment): una per recuperare un singolo tipo utilizzando un `id` o uno qualsiasi dei campi del tipo elencati nell'input; l'altra per recuperare più valori, filtrati utilizzando l'input per il tipo in questione.
- Aggiunta di tre mutazioni per ogni tipo: create, update e delete. Il tipo da eliminare viene specificato utilizzando un `id` o l'input per quel tipo. Queste mutazioni influiscono sui dati archiviati nel database Neptune.
- Aggiunta di due mutazioni per le connessioni: connect e delete. Queste prendono come input gli id nodo dei vertici di partenza e di arrivo utilizzati da Neptune; la connessione è rappresentata dagli archi nel database.

Il resolver riconosce query e mutazioni in base al relativo nome, ma è possibile personalizzarle come illustrato di seguito.

Di seguito è riportato il contenuto del file `TodoExample.source.graphql`:

```
type Todo {
  _id: ID! @id
  name: String
```

```
  description: String
  priority: Int
  status: String
  comments(filter: CommentInput, options: Options): [Comment] @relationship(type:
 "CommentEdge", direction: OUT)
  bestComment: Comment @relationship(type: "CommentEdge", direction: OUT)
  commentEdge: CommentEdge
}

type Comment {
  _id: ID! @id
  content: String
}

input Options {
  limit: Int
}

input TodoInput {
  _id: ID @id
  name: String
  description: String
  priority: Int
  status: String
}

type CommentEdge {
  _id: ID! @id
}

input CommentInput {
  _id: ID @id
  content: String
}

input Options {
  limit: Int
}

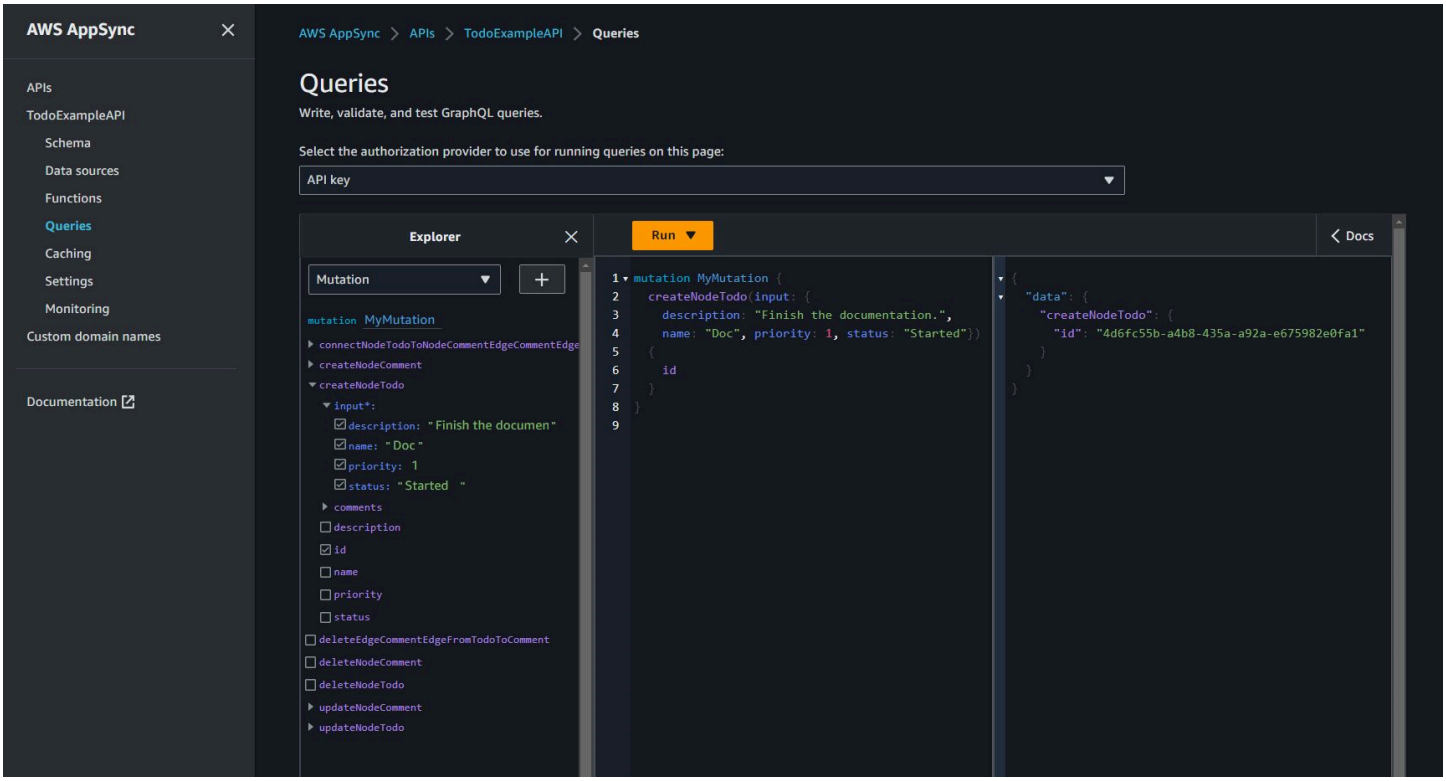
type Query {
  getNodeTodo(filter: TodoInput, options: Options): Todo
  getNodeTodos(filter: TodoInput): [Todo]
  getNodeComment(filter: CommentInput, options: Options): Comment
  getNodeComments(filter: CommentInput): [Comment]
```

```
}

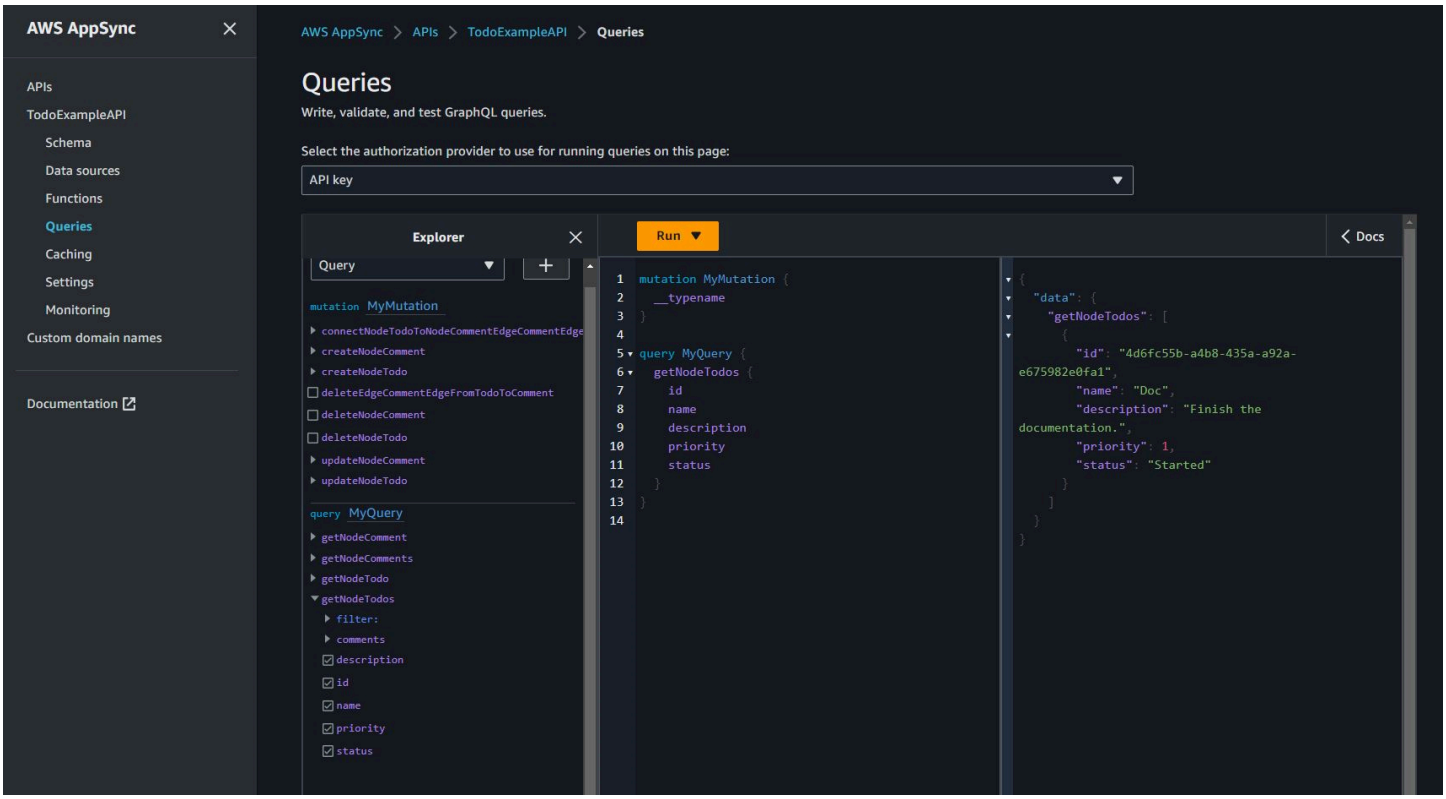
type Mutation {
  createNodeTodo(input: TodoInput!): Todo
  updateNodeTodo(input: TodoInput!): Todo
  deleteNodeTodo(_id: ID!): Boolean
  connectNodeTodoToNodeCommentEdgeCommentEdge(from_id: ID!, to_id: ID!): CommentEdge
  deleteEdgeCommentEdgeFromTodoToComment(from_id: ID!, to_id: ID!): Boolean
  createNodeComment(input: CommentInput!): Comment
  updateNodeComment(input: CommentInput!): Comment
  deleteNodeComment(_id: ID!): Boolean
}

schema {
  query: Query
  mutation: Mutation
}
```

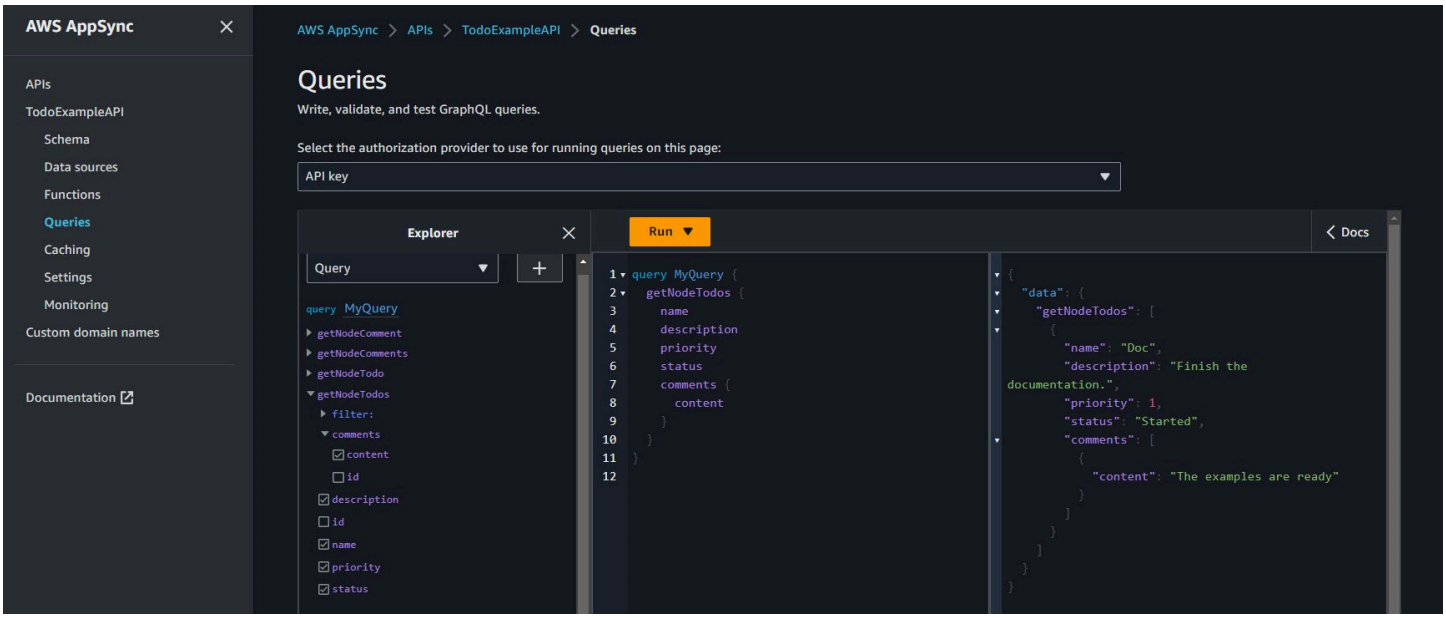
Ora puoi creare e interrogare i dati. Ecco un'istantanea della console AppSync Queries utilizzata per testare la nuova API GraphQL, denominata `TodoExampleAPI` in questo caso. Nella finestra centrale, lo strumento di esplorazione mostra un elenco di query e mutazioni da cui puoi scegliere una query, i parametri di input e i campi restituiti. Lo screenshot mostra la creazione di un tipo di nodo Todo utilizzando la mutazione `createNodeTodo`:



Lo screenshot mostra l'interrogazione di tutti i nodi Todo utilizzando la query `getNodeTodos`:



Dopo aver creato un Comment utilizzando `createNodeComment`, puoi usare la mutazione `connectNodeTodoToNodeCommentEdgeCommentEdge` per connetterli specificando i relativi id. Di seguito è riportata una query annidata per recuperare i nodi Todo e i relativi commenti allegati:



Se desideri apportare modifiche al file `TodoExample.source.graphql` come descritto nella sezione Uso delle direttive, puoi quindi utilizzare lo schema modificato come input ed eseguire nuovamente l'utilità. A questo punto, l'utilità modifica l'API GraphQL di conseguenza.

Uso delle direttive per uno schema GraphQL

Puoi iniziare da uno schema GraphQL che contiene già delle direttive, usando un comando come questo:

```
neptune-for-graphql \
  --input-schema-file (your GraphQL schema file with directives) \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (name for your new GraphQL API) \
  --create-update-aws-pipeline-neptune-endpoint  (empty Neptune database
 endpoint):(port number) \
  --output-resolver-query-https
```

Puoi modificare le direttive create dall'utilità o aggiungere direttive personalizzate a uno schema GraphQL. Di seguito sono descritti alcuni modi per usare le direttive:

Esecuzione dell'utilità in modo che non generi mutazioni

Per impedire all'utilità di generare mutazioni nell'API GraphQL, usa l'opzione `--output-schema-no-mutations` nel comando `neptune-for-graphql`.

Direttiva `@alias`

La direttiva `@alias` può essere applicata ai tipi o ai campi dello schema GraphQL. Essa esegue la mappatura di nomi diversi tra il database a grafo e lo schema GraphQL. La sintassi è:

```
@alias(property: (property name))
```

Nell'esempio seguente, `airport` è l'etichetta del nodo del database a grafo mappata al tipo GraphQL `Airport`, mentre `desc` è la proprietà del nodo del grafo mappata al campo `description` (consulta l'esempio per Air Routes):

```
type Airport @alias(property: "airport") {
  city: String
  description: String @alias(property: "desc")
```

```
}
```

Nota che la formattazione GraphQL standard richiama nomi di tipo Pascal-casing e nomi di campo camel-casing.

Direttiva `@relationship`

La direttiva `@relationship` mappa i tipi GraphQL annidati agli archi del database a grafo. La sintassi è:

```
@relationship(edgeType: (edge name), direction: (IN or OUT))
```

Di seguito è illustrato un esempio di comando:

```
type Airport @alias(property: "airport") {
  ...
  continentContainsIn: Continent @relationship(edgeType: "contains", direction: IN)
  countryContainsIn: Country @relationship(edgeType: "contains", direction: IN)
  airportRoutesOut(filter: AirportInput, options: Options): [Airport]
 @relationship(edgeType: "route", direction: OUT)
  airportRoutesIn(filter: AirportInput, options: Options): [Airport]
 @relationship(edgeType: "route", direction: IN)
}
```

Puoi trovare le direttive `@relationship` sia nell'esempio Todo che nell'esempio Air Routes.

Direttive `@graphQuery` e `@cypher`

Puoi definire query openCypher per risolvere un valore di campo, aggiungere query o aggiungere mutazioni. Ad esempio, puoi aggiungere un nuovo campo `outboundRoutesCount` al tipo `Airport` per contare le route in uscita:

```
type Airport @alias(property: "airport") {
  ...
  outboundRoutesCount: Int @graphQuery(statement: "MATCH (this)-[r:route]->(a) RETURN
 count(r)")
}
```

Di seguito è riportato un esempio con nuove query e mutazioni:

```
type Query {
  getAirportConnection(fromCode: String!, toCode: String!): Airport \
    @cypher(statement: \
      "MATCH (:airport{code: '$fromCode'})-[:route]->(this:airport)-[:route]-
>(:airport{code:'$toCode'})")
}

type Mutation {
  createAirport(input: AirportInput!): Airport @graphQuery(statement: "CREATE
 (this:airport {$input}) RETURN this")
  addRoute(fromAirportCode:String, toAirportCode:String, dist:Int): Route \
    @graphQuery(statement: \
     "MATCH (from:airport{code:'$fromAirportCode'}),
 (to:airport{code:'$toAirportCode'}) \
      CREATE (from)-[this:route{dist:$dist}]->(to) \
      RETURN this")
}
```

Nota: se ometti RETURN, il resolver presume che la parola chiave `this` sia l'ambito di restituzione.

Puoi anche aggiungere una query o una mutazione usando una query Gremlin:

```
type Query {
  getAirportWithGremlin(code:String): Airport \
    @graphQuery(statement: "g.V().has('airport', 'code', '$code').elementMap()")  #
 single node
  getAirportsWithGremlin: [Airport] \
    @graphQuery(statement: "g.V().hasLabel('airport').elementMap().fold()")       #
 list of nodes
  getCountriesCount: Int \
    @graphQuery(statement: "g.V().hasLabel('country').count()")                   #
 scalar
}
```

Al momento le query Gremlin sono limitate a quelle che restituiscono valori scalari, ovvero `elementMap()` per un singolo nodo o `elementMap().fold()` per un elenco di nodi.

Direttiva **`@id`**

La direttiva `@id` identifica il campo mappato all'entità del database a grafo `id`. I database a grafo come Amazon Neptune hanno sempre un `id` univoco per nodi e archi che viene assegnato durante le importazioni in blocco o generato automaticamente. Per esempio:

```
type Airport {
  _id: ID! @id
  city: String
  code: String
}
```

Nomi di tipo, query e mutazione riservati

L'utilità genera automaticamente query e mutazioni per creare un'API GraphQL funzionante. Il modello di questi nomi è riconosciuto dal resolver ed è riservato. Di seguito sono riportati alcuni esempi del tipo `Airport` e del tipo di connessione `Route`:

Il tipo `Options` è riservato.

```
input Options {
  limit: Int
}
```

I parametri `filter` e `options` della funzione sono riservati.

```
type Query {
  getNodeAirports(filter: AirportInput, options: Options): [Airport]
}
```

Il prefisso getNode dei nomi delle query è riservato, e anche i prefissi dei nomi delle mutazioni come`createNode`, `updateNode`, `deleteNode`, `connectNode`, `deleteNode`, `updateEdge` e `deleteEdge` sono riservati.

```
type Query {
  getNodeAirport(id: ID, filter: AirportInput): Airport
  getNodeAirports(filter: AirportInput): [Airport]
}

type Mutation {
  createNodeAirport(input: AirportInput!): Airport
  updateNodeAirport(id: ID!, input: AirportInput!): Airport
  deleteNodeAirport(id: ID!): Boolean
  connectNodeAirportToNodeAirportEdgeRout(from: ID!, to: ID!, edge: RouteInput!): Route
  updateEdgeRouteFromAirportToAirport(from: ID!, to: ID!, edge: RouteInput!): Route
  deleteEdgeRouteFromAirportToAirport(from: ID!, to: ID!): Boolean
```

```
}
```

Applicazione di modifiche allo schema GraphQL

Puoi modificare lo schema di origine GraphQL ed eseguire nuovamente l'utilità, ottenendo lo schema più recente dal database Neptune. Ogni volta che l'utilità rileva un nuovo schema nel database, genera un nuovo schema GraphQL.

Puoi anche modificare manualmente lo schema di origine GraphQL ed eseguire nuovamente l'utilità utilizzando come input lo schema di origine anziché l'endpoint del database Neptune.

Infine, puoi inserire le modifiche in un file utilizzando questo formato JSON:

```
[
  {
    "type": "(GraphQL type name)",
    "field": "(GraphQL field name)",
    "action": "(remove or add)",
    "value": "(value)"
  }
]
```

Per esempio:

```
[
  {
    "type": "Airport",
    "field": "outboundRoutesCountAdd",
    "action": "add",
    "value":"outboundRoutesCountAdd: Int @graphQuery(statement: \"MATCH (this)-
[r:route]->(a) RETURN count(r)\")"
  },
  {
    "type": "Mutation",
    "field": "deleteNodeVersion",
    "action": "remove",
    "value": ""
  },
  {
    "type": "Mutation",
    "field": "createNodeVersion",
    "action": "remove",
    "value": ""
```

```
  }
]
```

Quindi, quando esegui l'utilità su questo file utilizzando il parametro `--input-schema-changes-file` nel comando, l'utilità applica le modifiche tutte insieme.

Argomenti della riga di comando per l'utilità GraphQL

- **`--help, -h`**: restituisce alla console il testo della guida per l'utilità GraphQL.

- **`--input-schema`** ***`(schema text)`***: uno schema GraphQL, con o senza direttive, da usare come input.

- **`--input-schema-file`** ***`(file URL)`***: URL di un file contenente uno schema GraphQL da usare come input.

- **`--input-schema-changes-file`** ***`(file URL)`***: URL di un file contenente le modifiche da apportare a uno schema GraphQL. Se esegui più volte l'utilità su un database Neptune e in più modifichi manualmente lo schema di origine GraphQL (aggiungendo ad esempio una query personalizzata), le modifiche manuali andranno perse. Per evitare che ciò accada, inserisci le modifiche in un file dedicato e passalo utilizzando questo argomento.

  Il file delle modifiche usa il seguente formato JSON:

  ```
  [
    {
      "type": "(GraphQL type name)",
      "field": "(GraphQL field name)",
      "action": "(remove or add)",
      "value": "(value)"
    }
  ]
  ```

  Per ulteriori informazioni, consulta l'esempio Todo.

- **`--input-graphdb-schema` *`(schema text)`***: anziché eseguire l'utilità su un database Neptune, puoi esplicitare uno schema graphdb in formato testo da utilizzare come input. Uno schema graphdb ha un formato JSON come questo:

```
{
  "nodeStructures": [
    { "label":"nodelabel1",
      "properties": [
        { "name":"name1", "type":"type1" }
      ]
    },
    { "label":"nodelabel2",
      "properties": [
          { "name":"name2", "type":"type1" }
      ]
    }
  ],
  "edgeStructures": [
    {
      "label":"label1",
      "directions": [
        { "from":"nodelabel1", "to":"nodelabel2", "relationship":"ONE-ONE|ONE-MANY|
MANY-MANY"  }
      ],
      "properties": [
        { "name":"name1", "type":"type1" }
      ]
    }
  ]
}
```

- **`--input-graphdb-schema-file` *`(file URL)`***: anziché eseguire l'utilità su un database Neptune, puoi salvare uno schema graphdb in un file da utilizzare come input. Vedi `--input-graphdb-schema` qui sopra per un esempio del formato JSON per un file dello schema graphdb.

- **`--input-graphdb-schema-neptune-endpoint` *`(endpoint URL)`***: endpoint del database Neptune da cui l'utilità deve estrarre lo schema graphdb.

- **`--output-schema-file` *`(file name)`***: nome del file di output per lo schema GraphQL. Se non specificato, il valore predefinito è `output.schema.graphql`, a meno che non sia stato impostato un nome di pipeline utilizzando `--create-update-aws-pipeline-name`, nel qual caso il nome di file predefinito è *`(pipline name)`*`.schema.graphql`.

- **`--output-source-schema-file` *`(file name)`***: nome del file di output per lo schema GraphQL con direttive. Se non specificato, il valore predefinito è `output.source.schema.graphql`, a meno che non sia stato impostato un nome di pipeline utilizzando `--create-update-aws-pipeline-name`, nel qual caso il nome predefinito è *`(pipeline name)`*`.source.schema.graphql`.

- **`--output-schema-no-mutations`**: se questo argomento è presente, l'utilità non genera mutazioni nell'API GraphQL, ma solo query.

- **`--output-neptune-schema-file` *`(file name)`***: nome del file di output per lo schema graphdb di Neptune che l'utilità rileva. Se non specificato, il valore predefinito è `output.graphdb.json`, a meno che non sia stato impostato un nome di pipeline utilizzando `--create-update-aws-pipeline-name`, nel qual caso il nome di file predefinito è *`(pipeline name)`*`.graphdb.json`.

- **`--output-js-resolver-file` *`(file name)`***: nome del file di output per una copia del codice del resolver. Se non specificato, il valore predefinito è `output.resolver.graphql.js`, a meno che non sia stato impostato un nome di pipeline utilizzando `--create-update-aws-pipeline-name`, nel qual caso il nome del file è *`(pipeline name)`*`.resolver.graphql.js`.

  Questo file è compresso nel pacchetto di codice caricato nella funzione Lambda che esegue il resolver.

- **`--output-resolver-query-sdk`**: questo argomento specifica che la funzione Lambda dell'utilità deve interrogare Neptune utilizzando l'SDK per i dati Neptune, disponibile a partire dalla versione 1.2.1.0.R5 del motore Neptune (questa è l'impostazione predefinita). Tuttavia, se l'utilità rileva una versione precedente del motore Neptune, suggerisce di utilizzare invece l'opzione

HTTPS Lambda, che è possibile richiamare utilizzando l'argomento `--output-resolver-query-https`.

- **`--output-resolver-query-https`**: questo argomento specifica che la funzione Lambda dell'utilità deve interrogare Neptune utilizzando l'API HTTPS Neptune.

- **`--create-update-aws-pipeline`**— Questo argomento attiva la creazione delle AWS risorse da utilizzare per l'API GraphQL, tra cui l'API AppSync GraphQL e la Lambda che esegue il resolver.

- **`--create-update-aws-pipeline-name`** ***`(pipeline name)`***— Questo argomento imposta il nome della pipeline, ad esempio l'`pipeline-name`API AppSync o la `pipeline-name` funzione per la funzione Lambda. Se non viene specificato un nome, `--create-update-aws-pipeline` usa il nome del database `Neptune` .
- **`--create-update-aws-pipeline-region`** ***`(AWS region)`***: questo argomento imposta la regione AWS in cui viene creata la pipeline per l'API GraphQL. Se non è specificata, la regione predefinita è `us-east-1` o la regione in cui si trova il database Neptune, ricavata dall'endpoint del database.
- **`--create-update-aws-pipeline-neptune-endpoint`** ***`(endpoint URL)`***: questo argomento imposta l'endpoint del database Neptune utilizzato dalla funzione Lambda per interrogare il database. Se non è impostato, viene utilizzato l'endpoint impostato da `--input-graphdb-schema-neptune-endpoint`.
- **`--remove-aws-pipeline-name`** ***`(pipeline name)`***: questo argomento rimuove una pipeline creata utilizzando `--create-update-aws-pipeline`. Le risorse da rimuovere sono elencate in un file denominato *`(pipeline name)`*`.resources.json`.
- **`--output-aws-pipeline-cdk`**— Questo argomento attiva la creazione di un file CDK che può essere utilizzato per creare AWS le risorse per l'API GraphQL, tra cui l'API GraphQL e la funzione AppSync Lambda che esegue il resolver.
- **`--output-aws-pipeline-cdk-neptume-endpoint`** ***`(endpoint URL)`***: questo argomento imposta l'endpoint del database Neptune utilizzato dalla funzione Lambda per interrogare il database Neptune. Se non è impostato, viene utilizzato l'endpoint impostato da `--input-graphdb-schema-neptune-endpoint`.

- **`--output-aws-pipeline-cdk-name` *`(pipeline name)`*** — Questo argomento imposta il nome della pipeline per l' AppSync API e la funzione Lambda pipeline-name da utilizzare. Se non specificato, `--create-update-aws-pipeline` utilizza il nome del database Neptune.
- **`--output-aws-pipeline-cdk-region` *`(AWS region)`***: imposta la regione AWS in cui viene creata la pipeline per l'API GraphQL. Se non specificato, il valore predefinito è `us-east-1` o la regione in cui si trova il database Neptune, ricavata dall'endpoint del database.
- **`--output-aws-pipeline-cdk-file` *`(file name)`***: imposta il nome del file CDK. Se non è impostato, il valore predefinito è *`(pipeline name)`*`-cdk.js`.

Errori dei servizi Neptune

Amazon Neptune ha due set di errori diversi:

- Gli errori del motore basato su grafi che sono solo per gli endpoint del cluster database .
- Questi errori sono associati alle API per la creazione e la modifica di risorse Neptune con AWS SDK e la AWS Command Line Interface (AWS CLI).

Argomenti

Codici e messaggi di errore del motore basato su grafi

Gli endpoint di Amazon Neptune restituiscono gli errori standard per Gremlin e SPARQL quando si verificano.

Gli stessi endpoint possono anche restituire gli errori specifici per Neptune. In questa sezione vengono illustrati i messaggi di errore, i codici e le azioni consigliate per Neptune.

> **Note**
>
> Questi errori sono solo per gli endpoint del cluster database Neptune. Per le API per la creazione e la modifica di risorse Neptune con AWS SDK e la AWS CLI è disponibile un altro set di errori comuni. Per informazioni su questo tipo di errori, vedi the section called “Errori API”.

Formato errore del motore del grafo

I messaggi di errore di Neptune restituiscono un codice di errore HTTP pertinente e una risposta in formato JSON.

```
HTTP/1.1 400 Bad Request
```

```
x-amzn-RequestId: LDM6CJP8RMQ1FHKSC1RBVJFPNVV4KQNSO5AEMF66Q9ASUAAJG
Content-Type: application/x-amz-json-1.0
Content-Length: 465
Date: Thu, 15 Mar 2017 23:56:23 GMT

{
    "requestId": "0dbcded3-a9a1-4a25-b419-828c46342e47",
    "code": "ReadOnlyViolationException",
    "detailedMessage": "The request is rejected because it violates some read-only
 restriction, such as a designation of a replica as read-only."
}
```

Errori di query del motore del grafo

La tabella riportata di seguito contiene il codice di errore, il messaggio e lo stato HTTP.

Indica inoltre se è possibile riprovare a eseguire la richiesta. Generalmente, è possibile inviare una nuova richiesta se un nuovo tentativo può avere un esito positivo.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`AccessDeniedException`	403	No	Authentication or authorization failure.
`BadRequestException`	400	No	The request could not be completed.
`BadRequestException`	400	No	Request size exceeds max allowed value of 157286400 bytes.
`CancelledByUserException`	500	Yes	The request processing was cancelled by an authorized client.
`ConcurrentModificationException`	500	Yes	The request processing did not succeed due to a modification conflict. The

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
			client should retry the request.
`ConstraintViolationException`	400	Yes	The query engine discovered, during the execution of the request, that the completion of some operation is impossible without violating some data integrity constraints, such as persistence of in- and out-vertices while adding an edge. Such conditions are typically observed if there are concurrent modifications to the graph, and are transient. The client should retry the request.
`FailureByQueryException`	500	Yes	Calling fail() caused request processing to fail.
`InternalFailureException`	500	Yes	The request processing has failed.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`InvalidNumericDataException`	400	No	Invalid use of numeric data which cannot be represented in 64-bit storage size.
`InvalidParameterException`	400	No	An invalid or out-of-range value was supplied for some input parameter or invalid syntax in a supplied RDF file.
`MalformedQueryException`	400	No	The request is rejected because it contains a query that is syntactically incorrect or does not pass additional validation.
`MemoryLimitExceededException`	500	Yes	The request processing did not succeed due to lack of memory, but can be retried when the server is less busy.
`MethodNotAllowedException`	405	No	The request is rejected because the chosen HTTP method is not supported by the used endpoint.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`MissingParameterException`	400	No	A required parameter for the specified action is not supplied.
`QueryLimitExceededException`	500	Yes	The request processing did not succeed due to the lack of a limited resource, but can be retried when the server is less busy.
`QueryLimitException`	400	No	Size of query exceeds system limit.
`QueryTooLargeException`	400	No	The request was rejected because its body is too large.
`ReadOnlyViolationException`	400	No	The request is rejected because it violates some read-only restriction, such as a designation of a replica as read-only.
`ThrottlingException`	500	Yes	Rate of requests exceeds the maximum throughput. OK to retry.
`TimeLimitExceededException`	500	Yes	The request processing timed out.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`TooManyRequestsException`	429	Yes	The rate of requests exceeds the maximum throughput. OK to retry.
`UnsupportedOperationException`	400	No	The request uses a currently unsupported feature or construct.

Errori di autenticazione IAM

Questi errori sono specifici del cluster con l'autenticazione IAM abilitata.

La tabella riportata di seguito contiene il codice di errore, il messaggio e lo stato HTTP.

Neptune Service Error Code	HTTP status	Message
Incorrect IAM User/Policy	403	You do not have sufficient access to perform this action.
Incorrect or Missing Region	403	Credential should be scoped to a valid Region, not '*Regione*'.
Incorrect or Missing Service Name	403	Credential should be scoped to correct service: '`neptune-db `'.
Incorrect or Missing Host Header / Invalid Signature	403	The request signature we calculated does not match the signature you provided. Check your AWS Secret Access Key and signing method. Consult the service documentation for

Neptune Service Error Code	HTTP status	Message
		details. Host header is missing or hostname is incorrect.
Missing `X-Amz-Security-Token`	403	'`x-amz-security-token`' is named as a `SignedHeaders`, but it does not exist in the HTTP request
Missing Authorization Header	403	The request did not include the required authorization header, or it was malformed.
Missing Authentication Token	403	Missing Authentication Token.
Old Date	403	Signature expired: *20181011T213907Z* is now earlier than *20181011T213915Z* (*20181011T214415Z* - *5 min.*.)
Future Date	403	Signature not yet current: *20500224T213559Z* is still later than *20181108T225925Z* (*20181108T225425Z* + *5 min.*.)
Incorrect Date Format	403	Date must be in ISO-8601 'basic format'. Got '*data*'. See https://en.wikipedia.org/wiki/ISO_8601.
Unknown/Missing Access Key or Session Token	403	The security token included in the request is invalid.

Neptune Service Error Code	HTTP status	Message
Unknown/Missing Secret Key	403	The request signature we calculated does not match the signature you provided. Check your AWS Secret Access Key and signing method. Consult the service documentation for details. Host header is missing or hostname is incorrect.
`TooManyRequestsException`	429	The rate of requests exceeds the maximum throughput. OK to retry.

Codici e messaggi di errore dell'API di gestione del cluster di database

Questi errori di Amazon Neptune sono associati alle API per la creazione e la modifica di risorse Neptune con AWS SDK e la AWS CLI.

La tabella riportata di seguito contiene il codice di errore, il messaggio e lo stato HTTP.

Neptune Service Error Code	HTTP status	Message
`AccessDeniedException`	403	Non disponi dell'autorizzazione e di accesso sufficiente per eseguire questa operazione.
`IncompleteSignature`	400	La firma della richiesta non è conforme agli standard AWS.
`InternalFailure`	500	L'elaborazione della richiesta non è riuscita a causa di un errore, un'eccezione o un guasto interno sconosciuto.

Neptune Service Error Code	HTTP status	Message
`InvalidAction`	400	L'azione o l'operazione richiesta non è valida. Verifica che l'operazione sia digitata correttamente.
`InvalidClientTokenId`	403	Il certificato X.509 o l'ID chiave di accesso AWS forniti non sono presenti nei nostri record.
`InvalidParameterCombination`	400	Parametri che non possono essere utilizzati insieme sono stati utilizzati insieme.
`InvalidParameterValue`	400	Per il parametro di input è stato fornito un valore non valido o non compreso nell'intervallo valido.
`InvalidQueryParameter`	400	Per il parametro di input è stato fornito un valore non valido o non compreso nell'intervallo valido.
`MalformedQueryString`	400	La stringa di query contiene un errore di sintassi.
`MissingAction`	400	Nella richiesta manca un'operazione o un parametro obbligatorio.
`MissingAuthenticationToken`	403	La richiesta deve contenere un ID chiave di accesso AWS valido (registrato) o un certificato X.509.

Neptune Service Error Code	HTTP status	Message
`MissingParameter`	400	Un parametro richiesto per l'operazione specificata non è stato fornito.
`OptInRequired`	403	L'ID chiave di accesso AWS necessita di una sottoscrizione al servizio.
`RequestExpired`	400	La richiesta ha raggiunto il servizio più di 15 minuti dopo il date stamp della richiesta o più di 15 minuti dopo la data di scadenza della richiesta (ad esempio per URL prefirmati) oppure il date stamp della richiesta è più di 15 minuti nel futuro.
`ServiceUnavailable`	503	La richiesta non è riuscita a causa di un errore temporaneo del server.
`ThrottlingException`	500	La richiesta è stata negata a causa del throttling della richiesta.
`ValidationError`	400	L'input non riesce a soddisfare i vincoli specificati da un servizio AWS.

Messaggi di feed e di errore dello strumento di caricamento Neptune

I seguenti messaggi sono restituiti dall'endpoint `status` dello strumento di caricamento Neptune. Per ulteriori informazioni, consulta API dello stato get.

La tabella riportata di seguito contiene il codice di feed dello strumento di caricamento e una descrizione.

Error or Feed Code	Description
LOAD_NOT_STARTED	Il caricamento è stato registrato ma non avviato.
LOAD_IN_PROGRESS	Indica che il caricamento è in corso e specifica il numero di file attualmente caricati. Quando lo strumento di caricamento analizza un file, crea uno o più blocchi da caricare in parallelo. Poiché un singolo file può generare più blocchi, il numero di file incluso in questo messaggio è in genere inferiore al numero di thread utilizzati dal processo di caricamento in blocco.
LOAD_COMPLETED	Il caricamento è stato completato senza errori o con errori che rientrano in una soglia accettabile.
LOAD_CANCELLED_BY_USER	Il caricamento è stato annullato dall'utente.
LOAD_CANCELLED_DUE_TO_ERRORS	Il caricamento è stato annullato dal sistema a causa di errori.
LOAD_UNEXPECTED_ERROR	I caricamento non è riuscito con un errore imprevisto.
LOAD_FAILED	Caricamento non riuscito a causa di uno o più errori.
LOAD_S3_READ_ERROR	Il feed non è riuscito a causa di problemi intermittenti o transitori di connettività ad Amazon S3. Se uno qualsiasi dei feed riceve questo errore, lo stato generale del caricamento è impostato su LOAD_FAILED.

Error or Feed Code	Description
LOAD_S3_ACCESS_DENIED_ERROR	L'accesso è stato negato nel bucket S3. Se uno qualsiasi dei feed riceve questo errore, lo stato generale del caricamento è impostato su LOAD_FAILED.
LOAD_COMMITTED_W_WRITE_CONFLICTS	È stato eseguito il commit dei dati caricati con conflitti di scrittura non risolti. Lo strumento di caricamento proverà a risolvere i conflitti di scrittura in transazioni separate e ad aggiornare lo stato del feed con l'avanzamento del caricamento. Se lo stato finale del feed è LOAD_COMMITTED_W_WRITE_CONFLICTS, prova a riprendere il caricamento. È probabile che riesca senza conflitti di scrittura. Un conflitto di scrittura generalmente non è correlato a dati di input non validi, tuttavia la presenza di duplicati nei dati può aumentare la probabilità che si verifichino conflitti di scrittura.
LOAD_DATA_DEADLOCK	Load was automatically rolled back due to deadlock.
LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETED	Feed non riuscito perché il file è stato eliminato o aggiornato dopo l'inizio del caricamento.
LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED	La richiesta di caricamento non è stata eseguita perché il controllo delle dipendenze non riesce.
LOAD_IN_QUEUE	La richiesta di caricamento è stata accodata ed è in attesa di essere eseguita.
LOAD_FAILED_INVALID_REQUEST	Il caricamento non è riuscito perché la richiesta non era valida (ad esempio, l'origine/bucket specificato potrebbe non esistere o il formato del file non è valido).

Rilasci del motore per Amazon Neptune

Gli aggiornamenti dei rilasci del motore Amazon Neptune vengono resi disponibili periodicamente.

Puoi determinare il rilascio del motore attualmente installato utilizzando l'API instance-status o la console Neptune. Il numero di versione indica se si sta utilizzando una versione principale originale, una versione secondaria o una versione patch. Per ulteriori informazioni sulla numerazione delle release, vedere Numeri di versione del motore.

Per ulteriori informazioni sugli aggiornamenti in generale, consulta Manutenzione del cluster.

A partire dal rilascio del motore 1.3.0.0, le versioni del motore avranno la struttura mostrata nella tabella seguente. Il numero di versione secondario è quello che verrà valutato per l'elaborazione `AutoMinorVersionUpgrade`.

Versione	Versi del prod	Versi princ e	Versione secondari a	Versione della patch	Stato	Released	Fine del ciclo di vita	Aggiornam ento a:
1,31,0	1	3	1	0	attiva	2024-03-0 6	2025-11-3 0	N/D
1,30.0	1	3	0	0	attiva	2023-11-1 5	2025-11-3 0	1.3.1.0

La tabella seguente elenca tutte le versioni del motore a partire dalla 1.0.1.0, insieme alle informazioni sulla versione. end-of-life È possibile utilizzare le date in questa tabella per pianificare i cicli di test e aggiornamento.

Versione	Versior princip e	Versior second a	Stato	Released	Fine del ciclo di vita	Aggiornam ento a:
1.2.1.1	1.2	1.1	attiva	2024-03-11	2025-03-06	1,30,0
1.2.1.0	1.2	1	attiva	-08	2025-03-06	1,30,0

Versione	Versione principale	Versione secondaria	Stato	Released	Fine del ciclo di vita	Aggiornamento a:
1.2.0.2	1.2	0.2	attiva	2022-11-16	2025-03-06	1,30,0
1.2.0.1	1.2	0.1	attiva	2022-10-26	2025-03-06	1,30,0
1,20.0	1.2	0,0	attiva	2022-07-21	2025-03-06	1,30,0
1.1.1.0	1.1	1	attiva	2022-04-19	2024-10-31	1.2.1.0
1.1.0.0	1.1	0,0	attiva	2021-11-19	2024-10-31	1.1.1.0
1.0.5.1	1	5.1	obsoleta	2021-10-01	2023-01-30	1,10.0
1.0.5.0	1	5.0	obsoleta	2021-07-27	2023-01-30	1,10.0
1.0.4.2	1	4.2	obsoleta	2021-06-01	2023-01-30	1,10.0
1.0.4.1	1	4.1	obsoleta	2020-12-08	2023-01-30	1,10.0
1.0.4.0	1	4.0	obsoleta	2020-10-12	2023-01-30	1,10.0
1.0.3.0	1	3.0	obsoleta	2020-08-03	2023-01-30	1,10.0
1.0.2.2	1	2.2	obsoleta	2020-03-09	2022-07-29	1,03,0
1.0.2.1	1	2.1	obsoleta	2019-11-22	2022-07-29	1,03,0
1.0.2.0	1	2.0	obsoleta	2019-11-08	2020-05-19	1,03,0
1.0.1.2	1	1.2	obsoleta	2019-10-15	—	—
1,01.1	1	1.1	obsoleta	2019-08-13	—	—
1.0.1.0.*	1	1.0.*	obsoleta	02/07/2019 e prima	—	—

Pianificazione delle principali versioni del motore end-of-life

Le versioni del motore Neptune raggiungono quasi sempre la fine del loro ciclo di vita al termine di un trimestre di calendario. Le eccezioni si verificano solo quando sorgono importanti problemi di sicurezza o disponibilità.

Quando una versione del motore raggiunge la fine del ciclo di vita, ti verrà richiesto di aggiornare il database Neptune a una versione più recente.

In generale, le versioni del motore Neptune continueranno a essere disponibili come segue:

- Versioni secondarie del motore: le versioni secondarie del motore rimangono disponibili per almeno sei mesi dopo il rilascio.
- Versioni principali del motore: le versioni principali del motore rimangono disponibili per almeno 12 mesi dopo il rilascio.

Almeno 3 mesi prima che una versione del motore raggiunga la fine del ciclo di vita, AWS invierà una notifica e-mail automatica all'indirizzo e-mail associato al tuo AWS account e pubblicherà lo stesso messaggio sul tuo AWS Health Dashboard. In questo modo avrai tempo per pianificare e prepararti per l'aggiornamento.

Quando una versione del motore raggiunge la fine del ciclo di vita, non sarà più possibile creare nuovi cluster o istanze utilizzando tale versione e la scalabilità automatica non potrà più creare istanze utilizzando tale versione.

Una versione del motore che raggiunge effettivamente la fine del ciclo di vita verrà aggiornata automaticamente durante una finestra di manutenzione. Il messaggio che ti verrà inviato tre mesi prima della fine del ciclo di vita della versione del motore conterrà informazioni dettagliate sull'aggiornamento automatico, inclusa la versione implementata con l'aggiornamento automatico, l'impatto sui tuoi cluster database e le azioni consigliate.

> **Important**
>
> È tua responsabilità mantenere aggiornate le versioni del motore di database. AWS consiglia a tutti i clienti di aggiornare i propri database alla versione più recente del motore per usufruire delle più recenti misure di sicurezza, privacy e disponibilità. Se utilizzi il database su un motore o un software non supportato oltre la data di obsolescenza ("motore legacy"), aumenterà la probabilità del verificarsi di problemi di sicurezza, privacy e operativi, inclusi tempi di inattività.

Il funzionamento del database dell'utente su qualsiasi motore è soggetto all'Accordo che disciplina l'utilizzo dei AWS Servizi. I motori legacy non sono generalmente disponibili. AWS non fornisce più supporto per il Legacy Engine e AWS può porre limiti all'accesso o all'uso di qualsiasi Motore Legacy in qualsiasi momento, se AWS determina che il Legacy Engine rappresenta un rischio per la sicurezza o la responsabilità o un rischio di danno per i Servizi AWS, le sue Affiliate o terze parti. La decisione dell'utente di continuare a utilizzare il proprio contenuto in un motore legacy potrebbe comportare l'indisponibilità, il danneggiamento o l'irrecuperabilità del contenuto dell'utente. I database in esecuzione su un motore Legacy sono soggetti alle eccezioni dell'Accordo sul livello di servizio (SLA).
I DATABASE E IL SOFTWARE CORRELATO IN ESECUZIONE SU UN MOTORE LEGACY CONTENGONO BUG, ERRORI, DIFETTI E/O COMPONENTI DANNOSI.
DI CONSEGUENZA, E NONOSTANTE QUALSIASI DISPOSIZIONE CONTRARIA CONTENUTA NEL CONTRATTO O NEI TERMINI DI SERVIZIO, FORNISCE IL MOTORE LEGACY «COSÌ COM' AWS È».

Amazon Neptune Engine versione 1.3.1.0 (2024-03-06)

A partire dal 2024-03-06, la versione del motore 1.3.1.0 viene generalmente distribuita. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Note

La versione del motore 1.3.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.3.0.0 alla versione del motore 1.3.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.3`. Le versioni precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` o `neptune1.2` e tali gruppi di parametri non funzionano con la versione 1.3.0.0 e successive. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.

Miglioramenti in questo rilascio del motore

Miglioramenti generali

- Neptune ha migliorato l'avviso mostrato in profile/explain.
- Sono state rimosse le curve NIST EC obsolete dai gruppi denominati predefiniti utilizzati durante la negoziazione TLS. Le curve rimosse sono sect409k1, sect409r1 e sect571k1.

Miglioramenti a Gremlin

- Calcolo delle statistiche DFE migliorato per evitare NCU molto elevate di istanze Serverless.
- Miglioramento delle prestazioni di Gremlin per WITHIN.

Difetti corretti in questo rilascio del motore

Correzioni Gremlin

- Correzioni di bug relative all'ordinazione del piano di interrogazione del motore Gremlin DFE.
- Correzione di bug per le query Gremlin con un attraversamento opzionale, ad esempio per le query nel formato `g.v () .hasLabel ('person') .group () .by (id ()) .by (__.in ('friend') .id () .fold ()) `, in cui nessuna persona senza bordi amici veniva raggruppata.
- Correzione di bug per le query Gremlin quando il motore di query DFE fallisce con il modulatore CoalesceStep inside by ().
- Correzione di bug per le query Gremlin con TinkerPop sessione abilitata quando le query in una sessione falliscono anche quando tutte sono di sola lettura e si connettono a un'istanza del lettore.
- Correzione di un bug in cui IAM ARN non era presente nel registro di controllo per una richiesta iniziale di connessione websocket riuscita con successo per Gremlin.
- Coalesce Step, copertura con DFE. IdentifyStep
- Ottimizzazione del set di caratteristiche per interi piani DFE.

Correzioni apportate a openCypher

- Correzioni di bug nella clausola OpenCypher SET per consentire l'impostazione su un'espressione non variabile (ad esempio: match (n:test) set (caso in cui n.prop = 2 then n end) .prop = 3 return n.prop.

- Correzione di bug per le query OpenCypher non riuscite che coinvolgevano l'aggregazione e l'ordine per.
- Migliore UNWIND di elenchi di grandi dimensioni contenenti mappe statiche.
- Correzione di bug della query OpenCypher MERGE che utilizzava un ID personalizzato con valori duplicati.

Correzioni apportate a SPARQL

- È stato corretto un bug SPARQL relativo all'ambito delle variabili nei modelli opzionali.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster DB alla versione 1.3.1.0, assicurati che il tuo progetto sia compatibile con queste versioni in linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.6.2`
- Versione più recente di Gremlin supportata: `3.6.5`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento alla versione 1.3.1.0 del motore

Puoi eseguire l'aggiornamento a questo rilascio dal rilascio del motore 1.2.0.0 o successivi.

Aggiornamento a questo rilascio

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.1.0 \
    --allow-major-version-upgrade \
```

```
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Invece di `--apply-immediately`, puoi specificare `--no-apply-immediately`. Per eseguire un aggiornamento della versione principale, il allow-major-version-upgrade parametro è obbligatorio. Assicurati inoltre di includere la versione del motore onde evitare che il tuo motore venga aggiornato a una versione diversa.

Se il cluster utilizza un gruppo di parametri del cluster personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Analogamente, se alcune istanze del cluster utilizzano un gruppo di parametri del database personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team di AWS supporto è disponibile nei forum della community e tramite AWS Premium Support.

Motore Amazon Neptune versione 1.3.0.0 (15/11/2021)

A partire dal 15 novembre 2023, la versione del motore 1.3.0.0 viene implementata a livello generale. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Note**
>
> La versione del motore 1.3.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.3.0.0 alla versione del motore 1.3.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.3`. Le versioni precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` o `neptune1.2` e tali gruppi di parametri non funzionano con la versione 1.3.0.0 e successive. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.

Nuove caratteristiche in questo rilascio del motore

- È stata rilasciata l'API dei dati di Neptune.

  L'API di dati di Amazon Neptune fornisce supporto SDK per più di 40 operazioni sui dati di Neptune, incluse caricamento di dati, esecuzione di query, richiesta di dati e machine learning. Supporta tutti e tre i linguaggi di query Neptune (Gremlin, openCypher e SPARQL) ed è disponibile in tutti i linguaggi dell'SDK. Firma automaticamente le richieste delle API e semplifica enormemente l'integrazione di Neptune nelle applicazioni.
- È stato aggiunto il supporto per l'integrazione di OpenSearchServerless con Neptune.

Miglioramenti in questo rilascio del motore

Miglioramenti agli aggiornamenti del motore Neptune

Neptune ha cambiato la modalità di rilascio degli aggiornamenti del motore in modo da avere un maggiore controllo sul processo di aggiornamento. Anziché rilasciare patch per modifiche non irreversibili, Neptune ora rilascia versioni secondarie che possono essere controllate utilizzando

il campo dell'istanza `AutoMinorVersionUpgrade` e per le quali è possibile ricevere notifiche sottoscrivendo un abbonamento all'evento `RDS-EVENT-0156`.

Per ulteriori informazioni su queste modifiche, consulta Maintaining your Amazon Neptune DB Cluster.

Miglioramento della crittografia in transito

Neptune non supporta più i pacchetti di crittografia seguenti:

- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

Neptune supporta solo le seguenti suite di cifratura complessa con TLS 1.2:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`

Miglioramenti apportati a Gremlin

- È stato aggiunto il supporto nel motore DFE per le fasi Gremlin seguenti:
  - `FoldStep`
  - `GroupStep`
  - `GroupCountStep`
  - `TraversalMapStep`
  - `UnfoldStep`
  - `LabelStep`
  - `PropertyKeyStep`
  - `PropertyValueStep`
  - `AndStep`
  - `OrStep`
  - `ConstantStep`
  - `CountGlobalStep`

- Piani di query DFE di Gremlin ottimizzati per evitare scansioni complete dei vertici durante l'utilizzo della modulazione `by()`.
- Prestazioni notevolmente migliorate delle query a bassa cardinalità e bassa latenza.
- È stato aggiunto il supporto DFE per i predicati di filtro. TinkerPop `0r`
- Supporto DFE migliorato dell'attraversamento per i filtri sulla stessa chiave, per query come le seguenti:

```
g.withSideEffect("Neptune#useDFE", true)
 .V()
 .has('name', 'marko')
 .and(
   or(
     has('name', eq("marko")),
     has('name', eq("vardas"))
   )
 )
```

- Gestione degli errori migliorata per il passaggio `fail()`.

Miglioramenti di openCypher

- Prestazioni notevolmente migliorate delle query a bassa cardinalità e bassa latenza.
- Prestazioni di pianificazione delle query migliorate quando la query contiene molti tipi di nodi.
- Latenza ridotta di tutte le query VLP.
- Prestazioni migliorate rimuovendo i join di pipeline ridondanti per le query con un unico modello di nodo.
- Prestazioni migliorate per le query contenenti modelli multi-hop con cicli, come il seguente:

```
MATCH (n)-->()-->()-->(m)
RETURN n m
```

Miglioramenti di SPARQL

- Introdotto un nuovo operatore SPARQL: `PipelineHashIndexJoin`.
- Prestazioni di convalida URI migliorate per le query SPARQL.

- Prestazioni delle query di ricerca full-text SPARQL migliorate mediante la risoluzione in batch dei termini del dizionario.

Difetti corretti in questo rilascio del motore

Correzioni Gremlin

- È stato corretto un bug di Gremlin a causa del quale si verificava una perdita di transazioni durante il controllo dell'endpoint di stato delle query Gremlin per le query con predicati in attraversamenti figlio per fasi non elaborate in modo nativo nel motore DFE.
- È stato corretto un bug di Gremlin a causa del quale `valueMap()` non era ottimizzato nel motore DFE negli attraversamenti `by()`.
- È stato corretto un bug di Gremlin a causa del quale un'etichetta di fase collegata a `UnionStep` non veniva propagata all'ultimo elemento di percorso dei rispettivi attraversamenti figlio.
- Risolto un bug di Gremlin per cui una query non funzionava perché conteneva troppi TinkerPop passaggi e quindi non veniva eliminata.
- È stato corretto un bug di Gremlin a causa del quale veniva generato un `NullPointerException` nelle fasi `mergeV` e `mergeE`.
- È stato corretto un bug di Gremlin a causa del quale `order()` non ordinava correttamente gli output di stringa quando alcuni di essi contenevano un carattere di spazio.
- È stato risolto un problema di correttezza di Gremlin che si verificava durante l'elaborazione della fase `valueMap` nel motore DFE.
- È stato risolto un problema di correttezza di Gremlin che si verificava quando `GroupStep` o `GroupCountStep` era annidato in un attraversamento chiave.

Correzioni apportate a openCypher

- È stato risolto un bug di OpenCypher riguardante la gestione degli errori relativi ai caratteri NULL.
- È stato corretto un bug nella gestione delle transazioni Bolt di openCypher.

Correzioni apportate a SPARQL

- È stato corretto un bug SPARQL a causa del quale i valori all'interno delle funzioni ricorsive non venivano risolti correttamente.

- È stato corretto un bug di SPARQL che causava un peggioramento delle prestazioni quando veniva inserito un numero elevato di valori utilizzando la clausola `VALUES`.
- È stato corretto un bug di SPARQL a causa del quale una chiamata all'operatore `REGEX` su un valore letterale con tag di lingua non andava mai a buon fine.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster di database alla versione 1.3.0.0, accertarsi che il progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.6.2`
- Versione più recente di Gremlin supportata: `3.6.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.3.0.0

Puoi eseguire l'aggiornamento a questo rilascio dal rilascio del motore 1.2.0.0 o successivi.

Aggiornamento a questo rilascio

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
```

```
    --engine-version 1.3.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Invece di `--apply-immediately`, puoi specificare `--no-apply-immediately`. Per eseguire un aggiornamento della versione principale, il allow-major-version-upgrade parametro è obbligatorio. Assicurati inoltre di includere la versione del motore onde evitare che il tuo motore venga aggiornato a una versione diversa.

Se il cluster utilizza un gruppo di parametri del cluster personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Analogamente, se alcune istanze del cluster utilizzano un gruppo di parametri del database personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team di AWS supporto è disponibile nei forum della community e tramite AWS Premium Support.

Amazon Neptune Engine versione 1.2.1.1 (2024-03-11)

A partire dall'11 marzo 2024, la versione del motore 1.2.1.1 viene generalmente distribuita. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Note**
>
> La versione del motore 1.3.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.3.0.0 alla versione del motore 1.3.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.3`. Le versioni precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` o `neptune1.2` e tali gruppi di parametri non funzionano con la versione 1.3.0.0 e successive. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.

Miglioramenti in questo rilascio del motore

Miglioramenti generali

Neptune ha migliorato l'avviso mostrato in profile/explain.

Miglioramenti apportati a Gremlin

- È stato migliorato il calcolo delle statistiche DFE per evitare unità NCU molto elevate di istanze Serverless.
- Miglioramento delle prestazioni di Gremlin per WITHIN.

Difetti corretti in questo rilascio del motore

Correzioni Gremlin

- Correzioni di bug relative all'ordinazione del piano di interrogazione del motore Gremlin DFE.
- Correzione di un bug relativo all'errore out-of-memory Gremlin segnalato originariamente come. InternalFailureException
- Correzione di un bug per cui IAM ARN non era presente nel registro di controllo per una richiesta di connessione websocket iniziale riuscita.
- Correzione di bug per le query Gremlin con TinkerPop sessione abilitata quando le query in una sessione falliscono anche quando tutte sono di sola lettura e si connettono a un'istanza di lettura.

Correzioni apportate a openCypher

- Correzioni di bug nella clausola SET di OpenCypher per consentire l'impostazione su espressioni non variabili (ad esempio: match (n:test) set (caso in cui n.prop = 2 then n end) .prop = 3 return n.prop.
- Correzione di bug per le query OpenCypher non riuscite che coinvolgevano l'aggregazione e l'ordine per.
- Migliore UNWIND di elenchi di grandi dimensioni contenenti mappe statiche.
- Correzione di bug della query OpenCypher MERGE che utilizzava un ID personalizzato con valori duplicati.

Correzioni apportate a SPARQL

- Correzioni di bug nel pianificatore di query SPARQL DFE.
- Correzione di bug per SPARQL quando usato con le parole chiave BIND e OPTIONAL.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster DB alla versione 1.2.1.1, assicurati che il tuo progetto sia compatibile con queste versioni in linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.6.2`
- Versione più recente di Gremlin supportata: `3.6.2`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento alla versione 1.2.1.1 del motore

Puoi eseguire l'aggiornamento a questo rilascio dal rilascio del motore 1.2.0.0 o successivi.

Aggiornamento a questo rilascio

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Invece di `--apply-immediately`, puoi specificare `--no-apply-immediately`. Per eseguire un aggiornamento della versione principale, il allow-major-version-upgrade parametro è obbligatorio. Assicurati inoltre di includere la versione del motore onde evitare che il tuo motore venga aggiornato a una versione diversa.

Se il cluster utilizza un gruppo di parametri del cluster personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Analogamente, se alcune istanze del cluster utilizzano un gruppo di parametri del database personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team di AWS supporto è disponibile nei forum della community e tramite AWS Premium Support.

Motore Amazon Neptune versione 1.2.1.0 (08/03/2023)

A partire dall'8 marzo 2023, viene implementata a livello generale la versione del motore 1.2.1.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Note

Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.

- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Rilascio di patch successive per questa versione

- Rilascio: 1.2.1.0.R2 (02/05/2023)
- Rilascio: 1.2.1.0.R3 (13/06/2023)
- Rilascio: 1.2.1.0.R4 (10/08/2023)
- Rilascio: 1.2.1.0.R5 (02/09/2023)
- Rilascio: 1.2.1.0.R6 (12/09/2023)
- Rilascio: 1.2.1.0.R7 (06/10/2023)

Nuove caratteristiche in questa versione del motore

- Aggiunto il supporto per TinkerPop 3.6.2, che aggiunge molte nuove funzionalità di Gremlin come le nuove fasi `mergeV()`, `mergeE()`, `element()` e `fail()`. Le fasi `mergeV()` e `mergeE()` sono particolarmente importanti in quanto offrono un'opzione dichiarativa lungamente attesa per l'esecuzione di operazioni di tipo upsert, che dovrebbero semplificare notevolmente i modelli di codice esistenti e rendere Gremlin più facile da leggere. La versione 3.6.x ha anche aggiunto i predicati regex, un nuovo sovraccarico per la fase `property()` che richiede un `Map` e una revisione significativa del comportamento di modulazione di `by()` che diventa molto più coerente in tutte le fasi che lo utilizzano.

  Consulta il log delle modifiche di TinkerPop e la pagina dell'aggiornamento per informazioni sulle modifiche apportate nella versione 3.6 e sugli aspetti da considerare durante l'aggiornamento.

Se usi `fold().coalesce(unfold(), <mutate>)` per gli inserimenti condizionali, ti consigliamo di eseguire la migrazione alla nuova sintassi `mergeV/E()`, descritta qui e qui. Neptune usa un modello di blocco più restrittivo per `Merge` rispetto a quello per `Coalesce`, che può ridurre le eccezioni di modifica simultanea (CME).

Per ulteriori informazioni sulle nuove funzionalità disponibili in questo rilascio di TinkerPop, consulta il blog di Stephen Mallette, intitolato Exploring new features of Apache TinkerPop 3.6.x in Amazon Neptune.

- Aggiunto il supporto per i tipi di istanza R6i, basati sui processori Intel Xeon scalabili di terza generazione. Questi tipi rappresentano la soluzione ideale per carichi di lavoro che richiedono molta memoria e offrono prestazioni di calcolo/prezzo migliori fino al 15%, oltre a una larghezza di banda di memoria per vCPU fino al 20% superiore rispetto ai tipi di istanza R5 simili.
- Sono stati aggiunti gli endpoint API di riepilogo del grafo sia per i grafi di proprietà che per i grafi RDF, che consentono di ottenere rapidamente un report sul grafico desiderato.

  Per i grafi di proprietà (PG), l'API di riepilogo del grafo fornisce un elenco di sola lettura di etichette di nodi e archi, nonché di chiavi di proprietà, insieme al conteggio di nodi, archi e proprietà. Per i grafi RDF, fornisce un elenco di classi e chiavi di predicato, insieme al conteggio di quadruple, soggetti e predicati.

  Le seguenti modifiche sono state apportate con la nuova API di riepilogo del grafo:

  - È stata aggiunta una nuova azione dataplane GetGraphSummary.
  - È stato aggiunto un nuovo endpoint `rdf/statistics` in sostituzione dell'endpoint `sparql/statistics`, che ora è obsoleto.
  - Il nome del campo `summary` nella risposta di stato delle statistiche è stato modificato in `signatureInfo`, così da non confonderlo con le informazioni di riepilogo dei grafi. Le versioni precedenti del motore continuano a usare `summary` nella risposta JSON.
  - È stata modificata la precisione del campo `date` nella risposta di stato delle statistiche, da minuti a millisecondi. Il formato precedente era `2020-05-07T23:13Z` (precisione in minuti), mentre il nuovo formato è `2023-01-24T00:47:43.319Z` (precisione in millisecondi). Entrambi sono conformi allo standard ISO 8601, ma questa modifica può compromettere il codice esistente, a seconda di come viene analizzata la data.
  - È stato aggiunto un nuovo comando magic di riga `%statistics` nel Workbench che consente di recuperare le statistiche del motore DFE.

  - È stato aggiunto un nuovo comando magic di riga `%summary` nel Workbench che consente di recuperare le informazioni di riepilogo dei grafi.
- È stata aggiunta la registrazione di log delle query lente per registrare i log delle query che richiedono un tempo di esecuzione superiore a una soglia specificata. Puoi abilitare e controllare la registrazione di log delle query lente utilizzando due nuovi parametri dinamici, ossia neptune_enable_slow_query_log e neptune_slow_query_log_threshold.
- Aggiunto il supporto per due parametri dinamici, ossia due nuovi parametri di cluster: neptune_enable_slow_query_log e neptune_slow_query_log_threshold. Quando si apporta una modifica a un parametro dinamico, questa diventa immediatamente effettiva, senza richiedere il riavvio dell'istanza.
- È stata aggiunta una funzione removeKeyFromMap() openCypher specifica per Neptune che rimuove una chiave specificata da una mappa e restituisce la nuova mappa risultante.

Miglioramenti in questo rilascio del motore

- È stato esteso il supporto Gremlin DFE alle fasi `limit` con ambito locale.
- È stato aggiunto il supporto alla modulazione `by()` per `DedupGlobalStep` di Gremlin nel motore DFE.
- È stato aggiunto il supporto DFE per `SelectStep` e `SelectOneStep` di Gremlin.
- Miglioramenti delle prestazioni e correzioni della correttezza per vari operatori Gremlin, tra cui `repeat`, `coalesce`, `store` e `aggregate`.
- Prestazioni migliorate delle query openCypher che riguardano `MERGE` e `OPTIONAL MATCH`.
- Prestazioni migliorate delle query openCypher che riguardano `UNWIND` di un elenco di mappe di valori letterali.
- Prestazioni migliorate delle query openCypher che dispongono di un filtro `IN` per `id`. Per esempio:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- È stata aggiunta la possibilità di specificare l'IRI di base per le query SPARQL utilizzando l'istruzione BASE (consulta IRI di base predefinito per query e aggiornamenti).
- È stato ridotto il tempo di attesa per l'elaborazione del caricamento dei caricamenti in blocco edge-only su Gremlin e openCypher.

- I caricamenti in blocco ora riprendono in modo asincrono al riavvio di Neptune per evitare lunghi tempi di attesa causati da problemi di connettività di Amazon S3 prima dei tentativi non riusciti di riprendere i caricamenti.
- È stata migliorata la gestione delle query DESCRIBE di SPARQL che hanno l'hint di query describeMode impostato su `"CBD"` (descrizione limitata concisa) e che riguardano un numero elevato di nodi vuoti.

Difetti corretti in questa versione del motore

- È stato corretto un bug di openCypher per cui le query restituivano la stringa `"null"` anziché un valore nullo in Bolt and SPARQL-JSON.
- È stato corretto un bug di openCypher nella comprensione degli elenchi che generava un valore nullo anziché i valori forniti per gli elementi dell'elenco.
- È stato corretto un bug di openCypher a causa del quale i valori di byte non venivano serializzati correttamente.
- È stato corretto un bug di Gremlin in `UnionStep` che si verificava quando un input era un arco che attraversava verso un vertice all'interno di un attraversamento figlio.
- È stato corretto un bug di Gremlin che impediva la corretta propagazione di un'etichetta di fase associata a `UnionStep` all'ultima fase di ogni attraversamento figlio.
- È stato corretto un bug di Gremlin relativo alla fase `dedup` con etichette dopo una fase `dedup`, per cui le etichette allegate alla fase `repeat` non erano disponibili per essere utilizzate ulteriormente nella query.
- È stato corretto un bug di Gremlin a causa del quale la traduzione della fase `repeat` all'interno di una fase `union` non riusciva a causa di un errore interno.
- Sono stati risolti i problemi di correttezza di Gremlin per le query DFE con `limit` come attraversamento figlio di fasi di non unione tramite ricorso a TinkerPop. Sono coinvolte le query con una forma simile a questa:

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- È stato corretto un bug di SPARQL a causa del quale i modelli `SPARQL GRAPH` non consideravano il set di dati fornito da una clausola `FROM NAMED`.
- È stato corretto un bug di SPARQL a causa del quale le query `DESCRIBE` di SPARQL con alcune clausole `FROM` e/o `FROM NAMED` non utilizzavano sempre correttamente i dati del grafo predefinito

e talvolta generavano un'eccezione. Per informazioni, consultare Comportamento di SPARQL DESCRIBE rispetto al grafo predefinito.

- È stato corretto un bug di SPARQL in modo che venga restituito il messaggio di eccezione corretto quando i caratteri nulli vengono rifiutati.
- È stato corretto un bug di explain in SPARQL che interessava i piani contenenti un operatore PipelinedHashIndexJoin.
- È stato corretto un bug che causava la generazione di un errore interno quando veniva inviata una query che restituiva un valore costante.
- È stato risolto un problema relativo alla logica del rilevatore di deadlock che occasionalmente impediva al motore di rispondere.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.1.0, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.6.2`
- Versione più recente di Gremlin supportata: `3.6.2`
- Versione openCypher: `Neptune-9.0.20190305-1.1`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.1.0

È possibile eseguire manualmente l'aggiornamento a questo rilascio da qualsiasi rilascio precedente del motore Neptune superiore o uguale a 1.1.0.0.

> **Note**
>
> A partire dal rilascio 1.2.0.0 del motore, tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati utilizzati con le versioni del motore precedenti a `1.2.0.0` devono ora essere creati nuovamente utilizzando la famiglia di gruppi di parametri `neptune1.2`. Le versioni precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzioneranno con le versioni successive alla `1.2.0.0`. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.

L'aggiornamento a questo rilascio della versione principale non viene eseguito automaticamente.

Aggiornamento a questo rilascio

Amazon Neptune 1.2.1.0 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.1.0.R7 (06/10/2023)

A partire dal 6 ottobre 2023, la versione del motore 1.2.1.0.R7 viene implementata a livello generale. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Note

Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire

dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Difetti corretti in questa versione del motore

- È stato risolto un problema a causa del quale, in alcuni casi, una transazione non riuscita non veniva chiusa correttamente.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.1.0.R7, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.6.2`
- Versione più recente di Gremlin supportata: `3.6.2`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Aggiornamento a questo rilascio

Amazon Neptune 1.2.1.0.R7 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale,

questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.1.0.R6 (12/09/2023)

A partire dal 12 settembre 2023, viene implementata a livello generale la versione del motore 1.2.1.0.R6. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Note**
>
> Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:
>
> - Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia

di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.

- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Nuove caratteristiche in questa versione del motore

- È stata rilasciata l'API dei dati di Neptune.

  L'API dei dati di Amazon Neptune supporta gli SDK per il caricamento di dati, l'esecuzione di query, l'acquisizione di informazioni sui dati e l'esecuzione di operazioni di machine learning. Supporta i linguaggi di query Gremlin e openCypher in Neptune ed è disponibile in tutti i linguaggi degli SDK.

Firma automaticamente le richieste delle API e semplifica enormemente l'integrazione di Neptune nelle applicazioni.

Difetti corretti in questa versione del motore

- È stato corretto un bug che poteva causare picchi della CPU in caso di carichi elevati quando i log delle query lente erano abilitati.
- È stato corretto un bug di Gremlin a causa del quale l'aggiunta di un arco e delle relative proprietà seguiti da `inV()` o `outV()` generava una `InternalFailureException`.
- Sono stati risolti diversi problemi relativi al concatenamento dei ruoli IAM che in alcuni casi causavano una riduzione delle prestazioni dello strumento di caricamento in blocco.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.1.0.R6, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.6.2`
- Versione più recente di Gremlin supportata: `3.6.2`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Aggiornamento a questo rilascio

Amazon Neptune 1.2.1.0.R6 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale,

questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.1.0.R5 (02/09/2023)

A partire dal 2 settembre 2023, viene implementata a livello generale la versione del motore 1.2.1.0.R5. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Note**
>
> Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:
>
> - Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia

di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.

- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Nuove caratteristiche in questa versione del motore

- È stata rilasciata l'API dei dati di Neptune.

  L'API dei dati di Amazon Neptune supporta gli SDK per il caricamento di dati, l'esecuzione di query, l'acquisizione di informazioni sui dati e l'esecuzione di operazioni di machine learning. Supporta i linguaggi di query Gremlin e openCypher in Neptune ed è disponibile in tutti i linguaggi degli SDK.

Firma automaticamente le richieste delle API e semplifica enormemente l'integrazione di Neptune nelle applicazioni.

Difetti corretti in questa versione del motore

- È stato corretto un bug di Gremlin a causa del quale l'aggiunta di un arco e delle relative proprietà seguiti da `inV()` o `outV()` generava una `InternalFailureException`.
- Sono stati risolti diversi problemi relativi al concatenamento dei ruoli IAM che in alcuni casi causavano una riduzione delle prestazioni dello strumento di caricamento in blocco.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.1.0.R5, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.6.2`
- Versione più recente di Gremlin supportata: `3.6.2`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Aggiornamento a questo rilascio

Amazon Neptune 1.2.1.0.R5 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.1.0.R4 (10/08/2023)

A partire dal 10 agosto 2023, viene implementata a livello generale la versione del motore 1.2.1.0.R4. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Important**
>
> Le modifiche introdotte in questa versione del motore possono in alcuni casi causare un peggioramento delle prestazioni di caricamento in blocco. Di conseguenza, gli aggiornamenti a questo rilascio sono stati temporaneamente sospesi fino alla risoluzione del problema.

> **Note**
>
> Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:
>
> - Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si

esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.

- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Miglioramenti in questo rilascio del motore

- È stato aggiunto il supporto di GraphSON-1.0 per Gremlin. Per usare GraphSON-1.0, passa `Accept header` con un valore di:

```
application/vnd.gremlin-v1.0+json;types=false
```

Difetti corretti in questa versione del motore

- È stato corretto un bug di Gremlin a causa del quale si verificava una perdita di transazioni durante il controllo dell'endpoint di stato delle query Gremlin per le query con predicati in attraversamenti figlio per le fasi che non sono elaborate in modo nativo.
- È stato corretto un bug di openCypher nella gestione delle transazioni Bolt.
- È stato risolto un problema di simultaneità sul livello di archiviazione che poteva causare un arresto anomalo.
- È stato corretto un bug nei log delle query lente per assicurarsi che non siano attivi quando sono disabilitati.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.1.0.R4, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.6.2`
- Versione più recente di Gremlin supportata: `3.6.5`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.1.0.R4

Aggiornamento a questo rilascio

Amazon Neptune 1.2.1.0.R4 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.1.0.R3 (13/06/2023)

A partire dal 13 giugno 2023, viene implementata a livello generale la versione del motore 1.2.1.0.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Important**
>
> Le modifiche introdotte in questa versione del motore possono in alcuni casi causare un peggioramento delle prestazioni di caricamento in blocco. Di conseguenza, gli aggiornamenti a questo rilascio sono stati temporaneamente sospesi fino alla risoluzione del problema.

Note

Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile

aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Nuove caratteristiche in questa versione del motore

- Aggiunto il supporto per il caricamento in blocco tra più account utilizzando il concatenamento dei ruoli IAM.

Miglioramenti in questo rilascio del motore

- È stata migliorata la fase `fail()` di Gremlin volta a distinguere l'eccezione generata a partire da una `InternalFailureException` generica e a garantire che qualsiasi messaggio fornito dall'utente venisse propagato di nuovo al chiamante.
- Sono state migliorate le ottimizzazioni del motore di query di Gremlin per `store`, `aggregate`, `cap`, `limit` e `hasLabel`.
- Aggiunto il supporto per le funzioni trigonometriche di openCypher:
  - `acos()`
  - `asin()`
  - `atan()`
  - `atan2()`
  - `cos()`
  - `cot()`
  - `degrees()`
  - `pi()`
  - `radians()`
  - `sin()`
  - `tan()`
- Aggiunto il supporto per diverse funzioni di aggregazione openCypher:
  - `percentileDisc()`
  - `stDev()`
- Aggiunto il supporto per la funzione `epochmillis()` di openCypher che converte un `datetime` in `epochmillis`. Per esempio:

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

- Aggiunto il supporto per l'operatore modulo di openCypher (`%`).
- Aggiunto il supporto per lo strumento Static Debug Explain di openCypher.
- Aggiunto il supporto per la funzione `randomUUID()` di openCypher.
- Sono state migliorate le prestazioni di openCypher:
  - Sono stati migliorati il parser e il pianificatore di query.
  - È stato migliorato l'utilizzo della CPU nel motore DFE.
  - Sono state migliorate le prestazioni delle query contenenti più clausole di aggiornamento che riutilizzano le stesse variabili. Alcuni esempi sono:

    ```
    MERGE (n {name: 'John'})
      or
    MERGE (m {name: 'Jim'})
      or
    MERGE (n)-[:knows {since: 2023}]#(m)
    ```

  - Sono stati ottimizzati i piani di query per modelli di query multi-hop come:

    ```
    MATCH (n)-->()-->()-->(m)
    RETURN n m
    ```

  - Sono state migliorate le prestazioni dell'inserimento di elenchi e mappe tramite query parametrizzate. Per esempio:

    ```
    UNWIND $idList as id MATCH (n {`~id`: id})
    RETURN n.name
    ```

  - È stata migliorata l'esecuzione delle query contenenti `WITH` facendone una barriera appropriata.
  - È stata eseguita l'ottimizzazione per evitare la materializzazione ridondante dei valori nelle funzioni `Unfold` e di aggregazione.
- Sono state migliorate le prestazioni delle query SPARQL che contengono un numero elevato di input statici nella clausola `VALUES`, come:

```
SELECT ?n WHERE { VALUES (?name) { ("John") ("Jim") ... many values ... } ?n a ?
n_type . ?n ?name . }
```

- Sono state migliorate le prestazioni delle query CBD SPARQL.

Difetti corretti in questa versione del motore

- È stato corretto un bug di Gremlin a causa del quale le query di lunga durata con deep nesting causavano un elevato utilizzo della CPU e timeout delle query durante la fase di pianificazione delle query.
- È stato risolto un bug di Gremlin a causa del quale una `NullPointerException` non valida poteva essere generata quando si utilizzava `mergeV` e `mergeE`.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.1.0.R3, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.6.2`
- Versione più recente di Gremlin supportata: `3.6.2`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.1.0.R3

Aggiornamento a questo rilascio

Amazon Neptune 1.2.1.0.R3 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale,

questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.1.0.R2 (02/05/2023)

A partire dal 2 maggio 2023, viene implementata a livello generale la versione del motore 1.2.1.0.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Note**
>
> Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:
>
> - Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia

di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.

- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Miglioramenti in questo rilascio del motore

- È stato aggiunto un parametro `enableInterContainerTrafficEncryption` a tutte le API di Neptune ML, che puoi usare per abilitare e disabilitare la crittografia del traffico tra container durante i processi di formazione o di ottimizzazione degli iperparametri.
- È stato aggiunto il supporto multietichetta per le fasi `mergeV()` e `mergeE()` di Gremlin.

Difetti corretti in questa versione del motore

- È stato corretto un bug di openCypher a causa del quale le query di aggiornamento e restituzione non gestivano `orderBy`, `limit` o `skip` correttamente.
- È stato corretto un bug di openCypher che consentiva ai parametri contenuti in una richiesta di essere sovrascritti dai parametri contenuti in un'altra richiesta simultanea.
- È stato corretto un bug di openCypher in cui i log delle query lente non contenevano i tempi di query corretti.
- È stato corretto un bug di Gremlin a causa del quale poteva verificarsi una perdita di transazioni quando una query contenente `GroupCountStep` veniva inviata come una stringa.
- È stato corretto un bug di Gremlin a causa del quale le query WebSocket davano esito negativo quando erano abilitati i log delle query lente.
- È stato corretto un bug di Gremlin a causa del quale i log di debug storage-counter risultavano mancanti nei log delle query lente per le richieste WebSocket.
- Sono stati corretti diversi bug di Gremlin che coinvolgevano `mergeV()` e `mergeE()`.
- È stato corretto un bug di SPARQL a causa del quale i costi delle query al grafo denominato venivano stimati erroneamente, con conseguenti piani di query non ottimali ed errori di memoria insufficiente.
- È stato corretto un bug che influiva sull'autorizzazione per le query Gremlin e openCypher su un cluster abilitato per IAM.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.1.0.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.6.2`
- Versione più recente di Gremlin supportata: `3.6.2`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.1.0.R2

Aggiornamento a questo rilascio

Amazon Neptune 1.2.1.0.R2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.0.2 (20/11/2022)

A partire dal 20 novembre 2022, viene implementata a livello generale la versione del motore 1.2.0.2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Note

Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire

dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Rilascio di patch successive per questa versione

- Rilascio: 1.2.0.2.R2 (15/12/2022)
- Rilascio: 1.2.0.2.R3 (27/03/2023)
- Rilascio: 1.2.0.2.R4 (08/05/2023)
- Rilascio: 1.2.0.2.R5 (16/08/2023)
- Rilascio: 1.2.0.2.R6 (12/09/2023)

Nuove caratteristiche in questa versione del motore

- È stata introdotta l'inferenza induttiva in tempo reale per Gremlin in Neptune ML.
- È stata introdotta un'estensione openCypher che consente di specificare i valori ID personalizzati per le entità anziché gli UUID che Neptune genera altrimenti. La possibilità di assegnare ID personalizzati semplifica la migrazione a Neptune da Neo4j.

  Warning

  Questa estensione della specifica openCypher è incompatibile con le versioni precedenti, perché `~id` ora è considerato un nome di proprietà riservato. Se usi già `~id` come proprietà in dati e query, devi eseguire la migrazione della proprietà `~id` a una nuova chiave di proprietà prima di eseguire l'aggiornamento a questo rilascio.

- Sono state aggiunte svariate nuove modalità `DESCRIBE` SPARQL insieme agli hint di query per la relativa configurazione.

Miglioramenti in questo rilascio del motore

- Sono state migliorate le prestazioni di openCypher, in particolare per le query VLP.

- Sono state migliorate le prestazioni DFE per le query Gremlin con limiti non terminali, come:

```
g.withSideEffect('Neptune#useDFE',true).V().hasLabel('Student').limit(5).out('takesCourse')
```

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.0.2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.0.2

Aggiornamento a questo rilascio

Amazon Neptune 1.2.0.2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.0.2.R6 (12/09/2023)

A partire dal 12 settembre 2023, viene implementata a livello generale la versione del motore 1.2.0.2.R6. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Note

Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una

versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.

- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Difetti corretti in questa versione del motore

- È stato corretto un bug di SPARQL per cui l'operatore `REGEX` non dava mai esito positivo quando veniva chiamato su un valore letterale con tag di lingua.
- È stato risolto un problema che causava una riduzione delle prestazioni di caricamento in blocco.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.0.2.R6, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.5`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.0.2.R6

Se si esegue la versione del motore `1.2.0.2`, il cluster database Neptune verrà aggiornato automaticamente a questo rilascio di patch durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

Amazon Neptune 1.2.0.2.R6 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.0.2.R5 (16/08/2023)

A partire dal 16 agosto 2023, viene implementata a livello generale la versione del motore 1.2.0.2.R5. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Important

Le modifiche introdotte in questa versione del motore possono in alcuni casi causare un peggioramento delle prestazioni di caricamento in blocco. Di conseguenza, gli aggiornamenti a questo rilascio sono stati temporaneamente sospesi fino alla risoluzione del problema.

Note

Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.

- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Difetti corretti in questa versione del motore

- È stato corretto un bug di Gremlin a causa del quale `order()` non ordinava correttamente gli output di stringa quando alcuni di essi contenevano un carattere di spazio.
- È stato corretto un bug di Gremlin a causa del quale si verificava una perdita di transazioni durante il controllo dell'endpoint di stato delle query Gremlin per le query con predicati in attraversamenti figlio per le fasi che non sono elaborate in modo nativo.
- È stato corretto un bug di openCypher nella gestione delle transazioni Bolt.
- È stato risolto un problema di simultaneità sul livello di archiviazione che poteva causare un arresto anomalo.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.0.2.R5, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.5`

- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.0.2.R5

Se si esegue la versione del motore `1.2.0.2`, il cluster database Neptune verrà aggiornato automaticamente a questo rilascio di patch durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

Amazon Neptune 1.2.0.2.R5 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un

aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.0.2.R4 (08/05/2023)

A partire dall'8 maggio 2023, viene implementata a livello generale la versione del motore 1.2.0.2.R4. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Note

Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare

a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.

- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Difetti corretti in questa versione del motore

- È stato corretto un bug di SPARQL a causa del quale un numero elevato di valori inseriti tramite la clausola `VALUES` poteva portare a una riduzione delle prestazioni.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.0.2.R4, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.6`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.0.2.R4

Se si esegue la versione del motore `1.2.0.2`, il cluster database Neptune verrà aggiornato automaticamente a questo rilascio di patch durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

Amazon Neptune 1.2.0.2.R4 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.0.2.R3 (27 marzo 2023)

A partire dal 27 marzo 2023, viene implementata a livello generale la versione del motore 1.2.0.2.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Note

Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire

dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Miglioramenti in questo rilascio del motore

- Per i cluster database serverless, sono state modificate l'impostazione della capacità minima a 1,0 NCU e l'impostazione massima valida più bassa a 2,5 NCU. Per informazioni, consultare Dimensionamento della capacità in un cluster database Neptune Serverless.
- È stato aggiunto un parametro `enableInterContainerTrafficEncryption` a tutte le API di Neptune ML, che puoi usare per abilitare e disabilitare la crittografia del traffico tra container durante i processi di formazione o di ottimizzazione degli iperparametri.

Difetti corretti in questa versione del motore

- È stato corretto un bug di Gremlin a causa del quale non era possibile riconoscere `option(Predicate)` come sintassi Gremlin valida.
- È stato corretto un bug di Gremlin che impediva la corretta pulizia delle query se queste davano esito negativo perché contenevano troppe fasi.
- È stato risolto un problema di correttezza di Gremlin che interessava le query DFE con `limit` come attraversamento figlio di fasi di non unione tramite ricorso a TinkerPop. Un esempio di query di questo tipo è:

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- È stata risolta una potenziale perdita di transazioni di Gremlin che si verificava quando una query inviata come String contiene `GroupCountStep`.
- È stato corretto un bug di openCypher a causa del quale il valore del parametro non veniva sempre dedotto correttamente in un elenco o un elenco di mappe.
- È stato corretto un bug di openCypher a causa del quale le query di aggiornamento e restituzione non gestivano `orderBy`, `limit` o `skip` correttamente.
- È stato corretto un bug di openCypher che consentiva ai parametri contenuti in una richiesta di essere sovrascritti dai parametri contenuti in un'altra richiesta simultanea.

- È stato corretto un bug di SPARQL a causa del quale un numero elevato di valori inseriti in una clausola `VALUES` poteva portare a una riduzione delle prestazioni.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.0.2.R3, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.6`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.0.2.R3

Se si esegue la versione del motore `1.2.0.2`, il cluster database Neptune verrà aggiornato automaticamente a questo rilascio di patch durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

Amazon Neptune 1.2.0.2.R3 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.0.2.R2 (15/12/2022)

A partire dal 15 dicembre 2022, viene implementata a livello generale la versione del motore 1.2.0.2.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Note**
>
> Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:
>
> - Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.

- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Miglioramenti in questo rilascio del motore

- Prestazioni migliorate delle query openCypher che riguardano `MERGE` e `OPTIONAL MATCH`.
- Prestazioni migliorate delle query openCypher che riguardano `UNWIND` di un elenco di mappe di valori letterali.
- Prestazioni migliorate delle query openCypher che dispongono di un filtro `IN` per `id`. Per esempio:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Miglioramenti delle prestazioni e correzioni della correttezza per vari operatori Gremlin, tra cui `repeat`, `coalesce`, `store` e `aggregate`.

Difetti corretti in questa versione del motore

- È stato corretto un bug di openCypher per cui le query restituivano la stringa `"null"` anziché un valore nullo in Bolt and SPARQL-JSON.
- È stato corretto un bug di Gremlin che impediva la propagazione di un'etichetta di fase allegata a `UnionStep` all'ultimo elemento di percorso dei relativi attraversamenti figlio.
- È stato corretto un bug di Gremlin che impediva l'ottimizzazione di `valueMap()` nel caso di un attraversamento `by()` nel motore DFE.
- È stato corretto un bug di Gremlin per cui le query di lettura eseguite come parte di una transazione Gremlin più lunga non bloccavano le righe.
- È stato corretto un bug del log di audit che causava la registrazione di informazioni non necessarie e l'assenza di alcuni campi nei log.
- È stato corretto un bug del log di audit a causa del quale l'ARN IAM delle richieste HTTP a un cluster database abilitato per IAM non veniva registrato.
- È stato corretto un bug relativo alla cache di ricerca in modo da limitare la memoria incrementale utilizzata per le istanze di scrittura nella cache.
- È stato corretto un bug relativo alla cache di ricerca che comportava l'impostazione della modalità di sola lettura per la cache di ricerca in caso di errore delle istanze di scrittura.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.0.2.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.0.2.R2

Se si esegue la versione del motore `1.2.0.2`, il cluster database Neptune verrà aggiornato automaticamente a questo rilascio di patch durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

Amazon Neptune 1.2.0.2.R2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.0.1 (26/10/2022)

A partire dal 26 ottobre 2022, viene implementata a livello generale la versione del motore 1.2.0.1. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Note

Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire

dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Rilascio di patch successive per questa versione

- Rilascio di manutenzione: 1.2.0.1.R2 (13/12/2022)
- Rilascio di manutenzione: 1.2.0.1.R3 (27/09/2023)

Nuove caratteristiche in questa versione del motore

- È stata introdotto Amazon Neptune Serverless, una soluzione di dimensionamento automatico on demand che aumenta le dimensioni dei cluster database in modo da soddisfare gli aumenti della domanda di elaborazione per poi ridurre nuovamente le dimensioni quando la domanda diminuisce.

Miglioramenti in questo rilascio del motore

- Sono state migliorate le prestazioni delle query `order-by` Gremlin. Le query Gremlin con `order-by` al termine di un `NeptuneGraphQueryStep` ora utilizzano blocchi di dimensioni superiori per prestazioni migliori. Questo non si applica a `order-by` in un nodo interno (non root) del piano di query.
- Sono state migliorate le prestazioni delle query di aggiornamento Gremlin. I vertici e gli archi devono ora essere bloccati per impedirne l'eliminazione durante l'aggiunta di archi o proprietà. Questa modifica elimina i blocchi duplicati all'interno di una transazione, migliorando così le prestazioni.
- Sono state migliorate le prestazioni delle query Gremlin che utilizzano `dedup()` all'interno di una sottoquery `repeat()` tramite spostamento di `dedup` verso il basso fino al livello di esecuzione nativo.
- Sono stati aggiunti messaggi di errore intuitivi per gli errori di autenticazione IAM. Questi messaggi ora mostrano l'utente IAM o l'ARN del ruolo, l'ARN della risorsa e un elenco di azioni

non autorizzate per la richiesta. L'elenco delle azioni non autorizzate ti consente di capire cosa potrebbe mancare o essere esplicitamente vietato nella policy IAM che stai utilizzando.

Difetti corretti in questa versione del motore

- È stato corretto un bug di Gremlin a causa del quale l'utilizzo di `PartitionStrategy` dopo l'aggiornamento a TinkerPop 3.5 generava erroneamente un errore con il messaggio "PartitionStrategy non funziona con gli attraversamenti anonimi", che impediva l'esecuzione dell'attraversamento.
- È stato corretto un bug di correttezza di Gremlin che riguardava la traduzione di `WherePredicateStep`, a causa del quale il motore di query di Neptune generava risultati errati per le query che usano `where(P.neq('x'))` e le relative varianti.
- È stato corretto un bug di openCypher nella clausola `MERGE` che in alcuni casi causava la creazione di nodi e archi duplicati.
- È stato corretto un bug di SPARQL nella gestione delle query che contengono `(NOT) EXISTS` all'interno di una clausola `OPTIONAL`, a causa del quale in qualche caso mancavano alcuni risultati delle query.
- È stato corretto un bug dello strumento di caricamento in blocco che causava regressioni delle prestazioni in caso di ingenti carichi di inserimento.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.0.1, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.0.1

Aggiornamento a questo rilascio

Amazon Neptune 1.2.0.1 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.0.1.R3 (27/09/2023)

A partire dal 27 settembre 2023, viene implementata a livello generale la versione del motore 1.2.0.1.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Note

Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire

dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Miglioramenti in questo rilascio del motore

- È stato aggiunto un parametro `enableInterContainerTrafficEncryption` a tutte le API di Neptune ML, che puoi usare per abilitare e disabilitare la crittografia del traffico tra container durante i processi di formazione o di ottimizzazione degli iperparametri.
- Per i cluster database serverless, sono state modificate l'impostazione della capacità minima a 1,0 NCU e l'impostazione massima valida più bassa a 2,5 NCU. Consulta Dimensionamento della capacità in un cluster database Neptune Serverless *(((prima del rilascio, questa modifica deve essere applicata anche alla pagina dedicata a serverless)))*.

Difetti corretti in questa versione del motore

- È stato corretto un bug di openCypher a causa del quale le query di aggiornamento e restituzione non gestivano `orderBy`, `limit` o `skip` correttamente.
- È stato corretto un bug di openCypher che consentiva ai parametri contenuti in una richiesta di essere sovrascritti dai parametri contenuti in un'altra richiesta simultanea.
- È stato corretto un bug di openCypher nella gestione delle transazioni Bolt.
- Sono stati risolti i problemi di correttezza di Gremlin per le query DFE con `limit` come attraversamento figlio di fasi di non unione tramite ricorso a TinkerPop. Ad esempio, per query come questa:

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
 .select("a")
 .by(out()
 .limit(1))
```

- È stato corretto un bug di Gremlin a causa del quale una query aveva esito negativo perché conteneva troppe fasi di TinkerPop e quindi non veniva pulita.
- È stato corretto un bug di Gremlin a causa del quale `order()` non ordinava correttamente gli output di stringa quando alcuni di essi contenevano un carattere di spazio.
- È stato corretto un bug di Gremlin a causa del quale poteva verificarsi una perdita di transazioni quando una query veniva inviata come String e conteneva `GroupCountStep`.
- È stato corretto un bug di Gremlin a causa del quale si verificava una perdita di transazioni durante il controllo dell'endpoint di stato delle query Gremlin per le query con predicati in attraversamenti figlio per le fasi che non sono elaborate in modo nativo.
- È stato corretto un bug di Gremlin a causa del quale l'aggiunta di un arco e delle relative proprietà seguiti da `inV()` o `outV()` causava `InternalFailureException`.
- È stato risolto un problema di simultaneità nel livello di archiviazione.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.0.1.R3, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.6`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.0.1.R3

Se si esegue la versione del motore 1.2.0.1, il cluster database Neptune verrà aggiornato automaticamente a questo rilascio di patch durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

Amazon Neptune 1.2.0.1.R3 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.0.1.R2 (13/12/2022)

A partire dal 13 dicembre 2022, viene implementata a livello generale la versione del motore 1.2.0.1.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Note**
>
> Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Miglioramenti in questo rilascio del motore

- Sono state migliorate le prestazioni delle query openCypher che riguardano `UNWIND` di un elenco di mappe di valori letterali.
- Miglioramenti delle prestazioni e correzioni della correttezza per vari operatori Gremlin, tra cui `repeat`, `coalesce`, `store` e `aggregate`.

Difetti corretti in questa versione del motore

- È stato corretto un bug di openCypher per cui le query restituivano la stringa `"null"` anziché un valore nullo in Bolt and SPARQL-JSON.
- È stato corretto un bug del log di audit che causava la registrazione di informazioni non necessarie e l'assenza di alcuni campi nei log.
- È stato corretto un bug relativo alla cache di ricerca in modo da limitare la memoria incrementale utilizzata per le istanze di scrittura nella cache.
- È stato corretto un bug relativo alla cache di ricerca che comportava l'impostazione della modalità di sola lettura per la cache di ricerca in caso di errore delle istanze di scrittura.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.0.1.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.0.1.R2

Se si esegue la versione del motore 1.2.0.1, il cluster database Neptune verrà aggiornato automaticamente a questo rilascio di patch durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

Amazon Neptune 1.2.0.1.R2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.0.0 (21/07/2022)

A partire dal 21 luglio 2022, viene implementata a livello generale la versione del motore 1.2.0.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Note

Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire

dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Rilascio di patch successive per questa versione

- Rilascio: 1.2.0.0.R2 (14/10/2022)
- Rilascio: 1.2.0.0.R3 (15/12/2022)
- Rilascio: 1.2.0.0.R4 (29/09/2023)

Nuove caratteristiche in questa versione del motore

- Aggiunto il supporto per i database globali. Un database globale Neptune si estende su più Regioni AWS ed è costituito da un cluster database principale in una regione e fino a cinque cluster database secondari in altre regioni.
- Aggiunto il supporto per un controllo degli accessi più granulare nelle policy IAM di Neptune rispetto a quello disponibile in precedenza, sulla base delle azioni del piano dati. Si tratta di un cambiamento radicale in quanto le policy IAM esistenti basate sull'azione `connect` obsoleta devono essere adattate per utilizzare le azioni più granulari del piano dati. Per informazioni, consultare Tipi di policy IAM.
- Maggiore disponibilità delle istanze di lettura In precedenza, al riavvio di un'istanza di scrittura, anche tutte le istanze di lettura nel cluster Neptune venivano riavviate automaticamente. A partire dal rilascio 1.2.0.0 del motore, le istanze di lettura rimangono attive dopo il riavvio di un'istanza di scrittura, il che migliora la disponibilità delle istanze di lettura. Le istanze di lettura possono essere riavviate separatamente per rendere effettive le modifiche ai gruppi di parametri. Per informazioni, consultare Riavvio di un'istanza database in Amazon Neptune.
- È stato aggiunto un nuovo parametro del cluster database neptune_streams_expiry_days che consente di impostare il numero di giorni in cui i record di flusso vengono conservati sul server prima di essere eliminati. L'intervallo è compreso tra 1 e 90 e il valore predefinito è 7.

Miglioramenti in questo rilascio del motore

- Sono state migliorate le prestazioni di serializzazione Gremlin per le query ByteCode.
- Neptune ora elabora i predicati di testo utilizzando il motore DFE, per migliorare le prestazioni.
- Neptune ora elabora le fasi `limit()` di Gremlin utilizzando il motore DFE, compresi i limiti di attraversamento non terminali e figlio.
- È stata modificata la gestione DFE della fase `union()` di Gremlin per integrarlo con altre nuove funzionalità, il che significa che i nodi di riferimento vengono visualizzati nei profili di query come previsto.
- Le prestazioni di alcune costose operazioni join all'interno di DFE sono state migliorate fino a un fattore massimo di 5 tramite parallelizzazione.
- È stato aggiunto il supporto alla modulazione `by()` per `OrderGlobalStep order(global)` per il motore DFE di Gremlin.
- È stata aggiunta la visualizzazione dei valori statici inseriti nei dettagli esplicativi per DFE.
- Prestazioni migliorate durante la rimozione dei modelli duplicati.
- È stato aggiunto il supporto per la conservazione dell'ordine nel motore DFE di Gremlin.
- Sono state migliorate le prestazioni delle query Gremlin con filtri vuoti, come questi:

```
g.V().hasId(P.within([]))
```

```
g.V().hasId([])
```

- È stata migliorata la messaggistica di errore nei casi in cui una query SPARQL usa un valore numerico troppo grande affinché Neptune possa rappresentarlo internamente.
- Sono state migliorate le prestazioni per l'eliminazione dei vertici con archi associati tramite riduzione delle ricerche negli indici quando i flussi sono disabilitati.
- È stato esteso il supporto per DFE a più varianti della fase `has()`, in particolare a `hasKey()`, `hasLabel()` e ai predicati di intervallo per stringhe/URI all'interno di `has()`. Questo influisce su query come le seguenti:

```
// hasKey() on properties
g.V().properties().hasKey("name")
g.V().properties().has(T.key, TextP.startingWith("a"))
g.E().properties().hasKey("weight")
g.E().properties().hasKey(TextP.containing("t"))
```

```
// hasLabel() on vertex properties
g.V().properties().hasLabel("name")

// range predicates on ID and Label fields
g.V().has(T.label, gt("person"))
g.E().has(T.id, lte("(an ID value)"))
```

- È stata aggiunta una funzione `join()` di openCypher specifica per Neptune che concatena le stringhe di un elenco in un'unica stringa.
- Sono state aggiornate le policy gestite da Neptune per includere le autorizzazioni di accesso ai dati e le autorizzazioni per le nuove API di database globali.

Difetti corretti in questa versione del motore

- È stato corretto un bug a causa del quale una richiesta HTTP senza un tipo di contenuto specificato aveva automaticamente un esito negativo.
- È stato corretto un bug di SPARQL nel sistema di ottimizzazione delle query che impediva l'uso di una chiamata di servizio all'interno di una query.
- È stato corretto un bug di SPARQL nel parser Turtle RDF a causa del quale una particolare combinazione di dati Unicode causava un errore.
- È stato corretto un bug di SPARQL a causa del quale una particolare combinazione di clausole `GRAPH` e `SELECT` generava risultati delle query errati.
- È stato corretto un bug di Gremlin che causava un problema di correttezza per le query che utilizzavano qualsiasi fase di filtro all'interno di una fase di unione, come la seguente:

```
g.V("1").union(hasLabel("person"), out())
```

- È stato corretto un bug di Gremlin a causa del quale `count()` di `both().simplePath()` comportava il doppio del numero effettivo di risultati restituiti senza `count()`.
- È stato corretto un bug di openCypher a causa del quale il server generava un'eccezione di errore per mancata corrispondenza della firma per le richieste Bolt ai cluster con l'autenticazione IAM abilitata.
- È stato corretto un bug di openCypher a causa del quale una query che utilizzava HTTP keep-alive poteva essere chiusa erroneamente se inviata dopo una richiesta non riuscita.
- È stato corretto un bug di openCypher che poteva causare la generazione di un errore interno quando veniva inviata una query che restituiva un valore costante.

- È stato corretto un bug nei dettagli di spiegazione in modo che la sottoquery `Time(ms)` di DFE sommi correttamente i tempi della CPU degli operatori all'interno della sottoquery di DFE. Consulta il seguente di output esplicativo a titolo di esempio:

```
subQuery1
#################################################################################################
# ID # Out #1 # Out #2 # Name                   # Arguments                                    #
 Mode # Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
  ...
#################################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=...graph#336e.../graph_1 #
 -    # 1        # 1         # 1.00  # 0.38      #
#    #        #        #                       # coordinationTime(ms)=0.026        #
     #          #           #       #           #
#################################################################################################
  ...
subQuery=...graph#336e.../graph_1
#################################################################################################
# ID # Out #1 # Out #2 # Name                   # Arguments                                    #
 Mode # Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # DFESolutionInjection  # solutions=[?100 -> [-10^^<LONG>]] #
 -    # 0        # 1         # 0.00  # 0.04      #
#    #        #        #                       # outSchema=[?100]                 #
     #          #           #       #           #
#################################################################################################
# 1  # 3      # -      # DFERelationalJoin     # joinVars=[]                      #
 -    # 2        # 1         # 0.50  # 0.29      #
#################################################################################################
# 2  # 1      # -      # DFESolutionInjection  # outSchema=[]                    #
 -    # 0        # 1         # 0.00  # 0.01      #
#################################################################################################
# 3  # -      # -      # DFEDrain              # -                                #
 -    # 1        # 0         # 0.00  # 0.02      #
#################################################################################################
```

I tempi subQuery nell'ultima colonna della tabella più in basso si sommano fino a dare un risultato di 0,36 ms (`.04 + .29 + .01 + .02 = .36`). Quando si aggiunge il tempo di coordinamento per tale sottoquery (`.36 + .026 = .386`), si ottiene un risultato che è vicino al tempo subQuery registrato nell'ultima colonna della tabella più in alto, ossia `0.38` ms.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.0.0, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.0.0

Poiché si tratta di un rilascio principale del motore, non è previsto un aggiornamento automatico.

Puoi eseguire l'aggiornamento al rilascio `1.2.0.0` solo manualmente, a partire dall'ultimo rilascio di patch del rilascio 1.1.1.0 del motore. Prima di poter essere aggiornati a `1.2.0.0`, i rilasci precedenti del motore devono prima essere aggiornati all'ultimo rilascio di `1.1.1.0`.

Pertanto, prima di provare a eseguire l'aggiornamento a questo rilascio, verifica che sia in uso il rilascio di patch più recente del rilascio `1.1.1.0`. Se non ne hai la certezza, inizia eseguendo l'aggiornamento al rilascio di patch più recente di `1.1.1.0`.

Prima dell'aggiornamento, devi anche creare nuovamente tutti i gruppi di parametri del cluster database personalizzati che usavi con la versione precedente, utilizzando la famiglia di gruppi di parametri `neptune1.2`. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.

Se prima esegui l'aggiornamento al rilascio `1.1.1.0` e subito dopo a `1.2.0.0`, è possibile che si verifichi un errore come quello riportato di seguito:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente (consulta Gestione del cluster di database Amazon Neptune).

Aggiornamento a questo rilascio

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Invece di `--apply-immediately`, puoi specificare `--no-apply-immediately`. Per eseguire un aggiornamento di versione principale, è richiesto il parametro allow-major-version-upgrade. Assicurati inoltre di includere la versione del motore onde evitare che il tuo motore venga aggiornato a una versione diversa.

Se il cluster utilizza un gruppo di parametri del cluster personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Analogamente, se alcune istanze del cluster utilizzano un gruppo di parametri del database personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.0.0.R4 (29/09/2023)

A partire dal 29 settembre 2023, viene implementata a livello generale la versione del motore 1.2.0.0.R4. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Note

Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.

- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Miglioramenti in questo rilascio del motore

- È stato aggiunto un parametro `enableInterContainerTrafficEncryption` a tutte le API di Neptune ML, che puoi usare per abilitare e disabilitare la crittografia del traffico tra container durante i processi di formazione o di ottimizzazione degli iperparametri.
- Per i cluster database serverless, sono state modificate l'impostazione della capacità minima a 1,0 NCU e l'impostazione massima valida più bassa a 2,5 NCU. Consulta Dimensionamento della capacità in un cluster database Neptune Serverless *(((prima del rilascio, questa modifica deve essere applicata anche alla pagina dedicata a serverless)))*.

Difetti corretti in questa versione del motore

- È stato corretto un bug di openCypher a causa del quale le query di aggiornamento e restituzione non gestivano `orderBy`, `limit` o `skip` correttamente.
- È stato corretto un bug di openCypher che consentiva ai parametri contenuti in una richiesta di essere sovrascritti dai parametri contenuti in un'altra richiesta simultanea.
- È stato corretto un bug di openCypher nella gestione delle transazioni Bolt.

- Sono stati risolti i problemi di correttezza di Gremlin per le query DFE con `limit` come attraversamento figlio di fasi di non unione tramite ricorso a TinkerPop. Ad esempio, per query come questa:

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
 .select("a")
 .by(out()
 .limit(1))
```

- È stato corretto un bug di Gremlin a causa del quale una query aveva esito negativo perché conteneva troppe fasi di TinkerPop e quindi non veniva pulita.
- È stato corretto un bug di Gremlin a causa del quale `order()` non ordinava correttamente gli output di stringa quando alcuni di essi contenevano un carattere di spazio.
- È stato corretto un bug di Gremlin a causa del quale poteva verificarsi una perdita di transazioni quando una query veniva inviata come String e conteneva `GroupCountStep`.
- È stato corretto un bug di Gremlin a causa del quale si verificava una perdita di transazioni durante il controllo dell'endpoint di stato delle query Gremlin per le query con predicati in attraversamenti figlio per le fasi che non sono elaborate in modo nativo.
- È stato corretto un bug di Gremlin a causa del quale l'aggiunta di un arco e delle relative proprietà seguiti da `inV()` o `outV()` causava `InternalFailureException`.
- È stato risolto un problema di simultaneità nel livello di archiviazione.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.0.0.R4, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.6`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.0.0.R4

Se si esegue la versione del motore `1.2.0.0`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

Puoi eseguire l'aggiornamento al rilascio `1.2.0.0` solo manualmente, a partire dall'ultimo rilascio di patch del rilascio 1.1.1.0 del motore. Prima di poter essere aggiornati a `1.2.0.0`, i rilasci precedenti del motore devono prima essere aggiornati all'ultimo rilascio di `1.1.1.0`.

Se prima esegui l'aggiornamento al rilascio `1.1.1.0` e subito dopo a `1.2.0.0`, è possibile che si verifichi un errore come quello riportato di seguito:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Aggiornamento a questo rilascio

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
```

```
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Invece di `--apply-immediately`, puoi specificare `--no-apply-immediately`. Per eseguire un aggiornamento di versione principale, è richiesto il parametro allow-major-version-upgrade. Assicurati inoltre di includere la versione del motore onde evitare che il tuo motore venga aggiornato a una versione diversa.

Se il cluster utilizza un gruppo di parametri del cluster personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Analogamente, se alcune istanze del cluster utilizzano un gruppo di parametri del database personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.0.0.R3 (15/12/2022)

A partire dal 15 dicembre 2022, viene implementata a livello generale la versione del motore 1.2.0.0.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Note**
>
> Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Miglioramenti in questo rilascio del motore

- Prestazioni migliorate delle query openCypher che riguardano `MERGE` e `OPTIONAL MATCH`.
- Sono state migliorate le prestazioni delle query openCypher che riguardano `UNWIND` di un elenco di mappe di valori letterali.
- Prestazioni migliorate delle query openCypher che dispongono di un filtro `IN` per `id`. Per esempio:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Miglioramenti delle prestazioni e correzioni della correttezza per vari operatori Gremlin, tra cui `repeat`, `coalesce`, `store` e `aggregate`.

Difetti corretti in questa versione del motore

- È stato corretto un bug di openCypher per cui le query restituivano la stringa `"null"` anziché un valore nullo in Bolt and SPARQL-JSON.
- È stato corretto un bug di openCypher in modo da poter interpretare correttamente il tipo di parametro quando il valore è un elenco o un elenco di mappe.
- È stato corretto un bug del log di audit che causava la registrazione di informazioni non necessarie e l'assenza di alcuni campi nei log.
- È stato corretto un bug del log di audit a causa del quale l'ARN IAM delle richieste HTTP a un cluster database abilitato per IAM non veniva registrato.
- È stato corretto un bug relativo alla cache di ricerca in modo da limitare la memoria incrementale utilizzata per le istanze di scrittura nella cache.
- È stato corretto un bug relativo alla cache di ricerca che comportava l'impostazione della modalità di sola lettura per la cache di ricerca in caso di errore delle istanze di scrittura.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.0.0.R3, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`

- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.0.0.R3

Se si esegue la versione del motore `1.2.0.0`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

Puoi eseguire l'aggiornamento al rilascio `1.2.0.0` solo manualmente, a partire dall'ultimo rilascio di patch del rilascio 1.1.1.0 del motore. Prima di poter essere aggiornati a `1.2.0.0`, i rilasci precedenti del motore devono prima essere aggiornati all'ultimo rilascio di `1.1.1.0`.

Se prima esegui l'aggiornamento al rilascio `1.1.1.0` e subito dopo a `1.2.0.0`, è possibile che si verifichi un errore come quello riportato di seguito:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Aggiornamento a questo rilascio

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Invece di `--apply-immediately`, puoi specificare `--no-apply-immediately`. Per eseguire un aggiornamento di versione principale, è richiesto il parametro allow-major-version-upgrade. Assicurati inoltre di includere la versione del motore onde evitare che il tuo motore venga aggiornato a una versione diversa.

Se il cluster utilizza un gruppo di parametri del cluster personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Analogamente, se alcune istanze del cluster utilizzano un gruppo di parametri del database personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.2.0.0.R2 (14/10/2022)

A partire dal 14 ottobre 2022, viene implementata a livello generale la versione del motore 1.2.0.0.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Note**
>
> Se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0:

- Il rilascio del motore 1.2.0.0 ha introdotto un nuovo formato per i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati. Di conseguenza, se si esegue l'aggiornamento da una versione del motore precedente alla 1.2.0.0 alla versione del motore 1.2.0.0 o successiva, è necessario creare nuovamente tutti i gruppi di parametri personalizzati e i gruppi di parametri del cluster personalizzati esistenti utilizzando la famiglia di gruppi di parametri `neptune1.2`. I rilasci precedenti utilizzavano la famiglia di gruppi di parametri `neptune1` e tali gruppi di parametri non funzionano con il rilascio 1.2.0.0 e successivi. Per ulteriori informazioni, consulta Gruppi di parametri di Amazon Neptune.
- Il rilascio del motore 1.2.0.0 ha inoltre introdotto un nuovo formato per i log di annullamento. Di conseguenza, tutti i log di annullamento creati da una versione precedente del motore devono essere rimossi e la metrica di CloudWatch `UndoLogsListSize` deve scendere a zero prima di poter avviare aggiornamenti da una versione precedente alla 1.2.0.0. Se sono presenti troppi record del log di annullamento (200.000 o più) quando si tenta di avviare un aggiornamento, il tentativo di aggiornamento può raggiungere il timeout in attesa del completamento della rimozione dei log di annullamento.

  È possibile accelerare la velocità di rimozione aggiornando l'istanza di scrittura del cluster, ovvero dove si verifica la rimozione. Se si esegue questa operazione prima di provare a eseguire l'aggiornamento, è possibile ridurre il numero di log di annullamento prima dell'avvio. L'aumento delle dimensioni dell'istanza di scrittura a un tipo di istanza 24XL può aumentare la velocità di rimozione fino a oltre un milione di record all'ora.

  Se la metrica `UndoLogsListSize` di CloudWatch è eccessivamente grande, l'apertura di un caso di supporto può aiutarti a esplorare altre strategie per ridurne le dimensioni.
- Infine, nel rilascio 1.2.0.0 è stata introdotta una modifica radicale che influisce sul codice precedente che utilizzava il protocollo Bolt con l'autenticazione IAM. A partire dal rilascio 1.2.0.0, Bolt necessita di un percorso della risorsa per la firma IAM. In Java l'impostazione del percorso della risorsa ha un aspetto simile al seguente: `request.setResourcePath("/openCypher"));`. In altri linguaggi, è possibile aggiungere `/openCypher` all'URI dell'endpoint. Per esempi, consulta Utilizzo del protocollo Bolt.

Miglioramenti in questo rilascio del motore

- Sono state migliorate le prestazioni delle query `order-by` Gremlin. Le query Gremlin con `order-by` al termine di un `NeptuneGraphQueryStep` ora utilizzano blocchi di dimensioni superiori per prestazioni migliori. Questo non si applica a `order-by` in un nodo interno (non root) del piano di query.
- Sono state migliorate le prestazioni delle query di aggiornamento Gremlin. I vertici e gli archi devono ora essere bloccati per impedirne l'eliminazione durante l'aggiunta di archi o proprietà. Questa modifica elimina i blocchi duplicati all'interno di una transazione, migliorando così le prestazioni.
- Sono state migliorate le prestazioni delle query Gremlin che utilizzano `dedup()` all'interno di una sottoquery `repeat()` tramite spostamento di `dedup` verso il basso fino al livello di esecuzione nativo.
- È stato aggiunto l'hint per la query `Neptune#cardinalityEstimates` Gremlin. Se impostato su `false`, disabilita le stime di cardinalità.
- Sono stati aggiunti messaggi di errore intuitivi per gli errori di autenticazione IAM. Questi messaggi ora mostrano l'utente IAM o l'ARN del ruolo, l'ARN della risorsa e un elenco di azioni non autorizzate per la richiesta. L'elenco delle azioni non autorizzate ti consente di capire cosa potrebbe mancare o essere esplicitamente vietato nella policy IAM che stai utilizzando.

Difetti corretti in questa versione del motore

- È stato corretto un bug di correttezza di Gremlin che riguardava la traduzione di `WherePredicateStep`, a causa del quale il motore di query di Neptune generava risultati errati per le query che usano `where(P.neq('x'))` e le relative varianti.
- È stato corretto un bug di Gremlin a causa del quale l'utilizzo di `PartitionStrategy` dopo l'aggiornamento a TinkerPop 3.5 generava erroneamente un errore con il messaggio "PartitionStrategy non funziona con gli attraversamenti anonimi", che impediva l'esecuzione dell'attraversamento.
- Sono stati corretti diversi bug di Gremlin relativi al valore `joinTime` di un comando join finale e alle statistiche all'interno dei sottogruppi di `Project.ASK`.
- È stato corretto un bug di openCypher nella clausola `MERGE` che in alcuni casi causava la creazione di nodi e archi duplicati.

- È stato corretto un bug di transazione a causa del quale una sessione poteva inserire dati dei grafi ed eseguire il commit anche quando i corrispondenti inserimenti simultanei del dizionario venivano ripristinati.
- È stato corretto un bug dello strumento di caricamento in blocco che causava regressioni delle prestazioni in caso di ingenti carichi di inserimento.
- È stato corretto un bug di SPARQL nella gestione delle query che contengono `(NOT) EXISTS` all'interno di una clausola `OPTIONAL`, a causa del quale in qualche caso mancavano alcuni risultati delle query.
- È stato corretto un bug a causa del quale i driver potevano sembrare bloccati nei casi in cui le richieste venivano annullate a causa di un timeout prima dell'avvio della valutazione. Era possibile entrare in questo stato se tutti i thread di elaborazione delle query sul server venivano utilizzati mentre si verificavano dei timeout per gli elementi nella coda delle richieste. Poiché i timeout della coda delle richieste non inviavano messaggi immediatamente, al client sembrava che le risposte restassero in sospeso.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.2.0.0.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.2.0.0.R2

Se si esegue la versione del motore `1.2.0.0`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

Puoi eseguire l'aggiornamento al rilascio `1.2.0.0` solo manualmente, a partire dall'ultimo rilascio di patch del rilascio 1.1.1.0 del motore. Prima di poter essere aggiornati a `1.2.0.0`, i rilasci precedenti del motore devono prima essere aggiornati all'ultimo rilascio di `1.1.1.0`.

Se prima esegui l'aggiornamento al rilascio `1.1.1.0` e subito dopo a `1.2.0.0`, è possibile che si verifichi un errore come quello riportato di seguito:

```
   We're sorry, your request to modify DB cluster (cluster identifier) has failed.
   Cannot modify engine version because instance (instance identifier) is
   running on an old configuration. Apply any pending maintenance actions on the
 instance before
   proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Aggiornamento a questo rilascio

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Invece di `--apply-immediately`, puoi specificare `--no-apply-immediately`. Per eseguire un aggiornamento di versione principale, è richiesto il parametro allow-major-version-upgrade. Assicurati inoltre di includere la versione del motore onde evitare che il tuo motore venga aggiornato a una versione diversa.

Se il cluster utilizza un gruppo di parametri del cluster personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Analogamente, se alcune istanze del cluster utilizzano un gruppo di parametri del database personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.1.1.0 (19/04/2022)

A partire dal 19 aprile 2022, viene implementata a livello generale la versione del motore 1.1.1.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Important**
>
> L'aggiornamento a questo rilascio del motore da una versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
> Per completare correttamente l'aggiornamento, ogni sottorete in ciascuna zona di disponibilità (AZ) deve avere almeno un indirizzo IP disponibile per ogni istanza di Neptune. Ad esempio, se ci sono un'istanza di scrittura e due istanze di lettura nella sottorete 1 e due istanze di

lettura nella sottorete 2, la sottorete 1 deve avere almeno 3 indirizzi IP liberi e la sottorete 2 deve avere almeno 2 indirizzi IP liberi prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per alcuni minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Rilascio di patch successive per questa versione

- Rilascio di manutenzione: 1.1.1.0.R2 (16/05/2022)
- Rilascio: 1.1.1.0.R3 (07/06/2022)
- Rilascio: 1.1.1.0.R4 (23/06/2022)
- Rilascio: 1.1.1.0.R5 (21/07/2022)
- Rilascio: 1.1.1.0.R6 (23/09/2022)
- Rilascio: 1.1.1.0.R7 (23/01/2023)

Nuove caratteristiche in questa versione del motore

- Il linguaggio di query openCypher è ora disponibile a livello generale per l'uso in produzione.

  > **Warning**
  >
  > In questo rilascio è presente una modifica radicale per il codice che usa openCypher con autenticazione IAM. Nell'anteprima di Neptune per openCypher, la stringa host nella firma IAM includeva il protocollo, come `bolt://`, ad esempio:
  >
  > ```
  > "Host":"bolt://(host URL):(port)"
  > ```
  >
  > A partire da questo rilascio del motore, il protocollo deve essere omesso:
  >
  > ```
  > "Host":"(host URL):(port)"
  > ```
  >
  > Per esempi, consulta Utilizzo del protocollo Bolt.

- Aggiunto il supporto per TinkerPop `3.5.2`. Tra le modifiche inserite in questa versione vi sono la compatibilità con le transazioni remote e il supporto di bytecode per le sessioni (utilizzando `g.tx`) e l'aggiunta della funzione `datetime()` al linguaggio Gremlin.

  > **Warning**
  >
  > Sono state introdotte diverse modifiche radicali in TinkerPop 3.5.0, 3.5.1 e 3.5.2 che potrebbero influire sul codice Gremlin. Ad esempio, non è più possibile utilizzare gli attraversamenti generati da GraphTraversalSource come figlio, come questo: `g.V().union(identity(), g.V())`.
  > Ora invece, viene utilizzato un attraversamento anonimo come questo: `g.V().union(identity(), __.V())`.

- Aggiunto il supporto per le chiavi di condizione globali AWS da utilizzare nelle policy di accesso ai dati IAM che controllano l'accesso ai dati archiviati in Neptune, un cluster database Neptune.
- Il motore di query Neptune DFE è ora disponibile a livello generale per l'uso in produzione con il linguaggio di query openCypher, ma non ancora per le query Gremlin e SPARQL. Adesso può essere abilitato utilizzando il relativo parametro di istanza neptune_dfe_query_engine anziché il parametro della modalità di laboratorio.

Miglioramenti in questo rilascio del motore

- Sono state aggiunte nuove funzionalità a openCypher, come il supporto per le query parametrizzate, la memorizzazione nella cache dell'albero sintattico astratto (AST) per le query parametrizzate, miglioramenti del percorso a lunghezza variabile (VLP), oltre a nuovi operatori e clausole. Per scoprire il livello attuale di supporto linguistico, consulta Conformità alle specifiche OpenCypher in Amazon Neptune.
- Sono stati apportati miglioramenti significativi alle prestazioni di openCypher per semplici carichi di lavoro in lettura e scrittura, con conseguente aumento della velocità di trasmissione effettiva rispetto al rilascio 1.1.0.0.
- Sono state rimosse le limitazioni bidirezionali e di profondità di openCypher per la gestione dei percorsi a lunghezza variabile.
- È stato completato il supporto nel motore DFE per i predicati `within` e `without` di Gremlin, inclusi i casi in cui vengono combinati con altri operatori di predicati. Per esempio:

```
g.V().has('age', within(12, 15, 18).or(gt(30)))
```

- È stato esteso il supporto nel motore DFE per la fase `order` di Gremlin quando l'ambito è globale (ovvero non `order(local)`) e quando i modulatori `by()` non vengono utilizzati. Ad esempio, questa query ora avrebbe il supporto DFE:

```
g.V().values("age").order()
```

- È stato aggiunto un campo `isLastOp` al formato di risposta del log delle modifiche apportate ai flussi Neptune, per indicare che un record è l'ultima operazione della relativa transazione.
- Sono state significativamente migliorate le prestazioni della registrazione di log di audit ed è stata ridotta la latenza quando la registrazione di log di audit è abilitata.
- Le query HTTP e bytecode WebSocket di Gremlin sono state convertite in un formato leggibile dall'utente nei log di audit. Le query possono ora essere copiate direttamente dai log di audit per essere eseguite nei notebook Neptune e altrove. Nota: questa modifica all'attuale formato dei log di audit rappresenta un cambiamento radicale.

Difetti corretti in questa versione del motore

- È stato corretto un raro bug di Gremlin per cui non veniva restituito alcun risultato quando si utilizzavano in combinazione fasi `filter()` e `count()` annidate, come nella seguente query:

```
g.V("1").filter(out("knows")
        .filter(in("knows")
        .hasId("notExists"))
        .count())
```

- È stato corretto un bug di Gremlin in cui veniva restituito un errore quando si utilizzava un vertice archiviato tramite una fase di aggregazione negli attraversamenti `to()` o `from()` congiuntamente a una fase `addE`. Un esempio di query di questo tipo è:

```
g.V("id").aggregate("v").out().addE().to(select("v").unfold()))
```

- È stato corretto un bug di Gremlin a causa del quale la fase `not` dava esito negativo nei casi limite in cui si utilizzava il motore DFE. Per esempio:

```
g.V().not(V())
```

- È stato corretto un bug di Gremlin a causa del quale i valori `sideEffect` non erano disponibili all'interno degli attraversamenti `to()` e `from()`.
- È stato corretto un bug che occasionalmente causava un ripristino rapido per innescare un failover dell'istanza.
- È stato corretto un bug dello strumento di caricamento in blocco a causa del quale una transazione non riuscita non veniva chiusa prima dell'inizio del successivo processo di caricamento.
- È stato corretto un bug dello strumento di caricamento in blocco a causa del quale una condizione di memoria insufficiente poteva causare un arresto anomalo del sistema.
- È stato aggiunto un nuovo tentativo per correggere un bug dello strumento di caricamento in blocco a causa del quale tale strumento non aspettava abbastanza a lungo prima che le credenziali IAM diventassero disponibili dopo un failover.
- È stato corretto un bug a causa del quale la cache interna delle credenziali non veniva cancellata correttamente per gli endpoint non sottoposti a query, come l'endpoint `status`.
- È stato corretto un bug relativo ai flussi per garantire il corretto ordinamento dei numeri di sequenza del commit dei flussi.
- È stato risolto un bug a causa del quale le connessioni a lunga durata venivano interrotte prima di dieci giorni nei cluster abilitati per IAM.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.1.1.0, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.1.1.0

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio. Nota: le versioni precedenti a quella principale del motore (1.1.0.0) impiegheranno più tempo per effettuare l'aggiornamento a questo rilascio.

L'aggiornamento a questo rilascio non viene eseguito automaticamente.

Aggiornamento a questo rilascio

Important

L'aggiornamento a questo rilascio del motore da qualsiasi versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per circa 6 minuti.
Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.

Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine neptune \
    --engine-version 1.1.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine neptune ^
    --engine-version 1.1.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Invece di `--apply-immediately`, puoi specificare `--no-apply-immediately`. Per eseguire un aggiornamento di versione principale, è richiesto il parametro allow-major-version-upgrade. Assicurati

inoltre di includere la versione del motore onde evitare che il tuo motore venga aggiornato a una versione diversa.

Se il cluster utilizza un gruppo di parametri del cluster personalizzato, assicurati di includere questo parametro per specificarlo:

```
--db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Analogamente, se alcune istanze del cluster utilizzano un gruppo di parametri del database personalizzato, assicurati di includere questo parametro per specificarlo:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come

lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.1.1.0.R7 (23/01/2023)

A partire dal 23 gennaio 2023, viene implementata a livello generale la versione del motore 1.1.1.0.R7. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Important**
>
> L'aggiornamento a questo rilascio del motore da una versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.

Per completare correttamente l'aggiornamento, ogni sottorete in ciascuna zona di disponibilità (AZ) deve avere almeno un indirizzo IP disponibile per ogni istanza di Neptune. Ad esempio, se ci sono un'istanza di scrittura e due istanze di lettura nella sottorete 1 e due istanze di lettura nella sottorete 2, la sottorete 1 deve avere almeno 3 indirizzi IP liberi e la sottorete 2 deve avere almeno 2 indirizzi IP liberi prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per alcuni minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Miglioramenti in questo rilascio del motore

- Prestazioni migliorate delle query openCypher che riguardano `MERGE` e `OPTIONAL MATCH`.
- Prestazioni migliorate delle query openCypher che riguardano `UNWIND` di un elenco di mappe di valori letterali.
- Prestazioni migliorate delle query openCypher che dispongono di un filtro `IN` per `id`. Per esempio:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Miglioramenti delle prestazioni e correzioni della correttezza per vari operatori Gremlin, tra cui `repeat`, `coalesce`, `store` e `aggregate`.

Difetti corretti in questa versione del motore

- È stato corretto un bug di openCypher a causa del quale una richiesta che utilizzava HTTP keep-alive poteva essere chiusa erroneamente se inviata dopo una richiesta non riuscita.
- È stato corretto un bug di openCypher a causa del quale il tipo di parametro non veniva sempre interpretato correttamente per un elenco o un elenco di mappe.
- È stato corretto un bug di openCypher per cui le query restituivano la stringa `"null"` anziché un valore nullo in Bolt and SPARQL-JSON.
- Sono stati corretti i codici di errore e i messaggi di errore openCypher per gli errori di timeout delle query e gli errori di memoria insufficiente.
- È stato corretto un bug di Gremlin che impediva l'ottimizzazione di `valueMap()` nel caso di un attraversamento `by()` nel motore DFE.
- È stato risolto un problema relativo alla logica del rilevatore di deadlock che occasionalmente impediva al motore di rispondere.
- È stato corretto un bug del log di audit che causava la registrazione di informazioni non necessarie e l'assenza di alcuni campi nei log.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.1.1.0.R7, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.3`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.1.1.0.R7

Se si esegue la versione del motore `1.1.1.0`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

Important

L'aggiornamento a questo rilascio del motore da qualsiasi versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per circa 6 minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.1.1.0.R6 (23/09/2022)

A partire dal 23 gennaio 2023, viene implementata a livello generale la versione del motore 1.1.1.0.R6. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Important**
>
> L'aggiornamento a questo rilascio del motore da una versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database.

Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
Per completare correttamente l'aggiornamento, ogni sottorete in ciascuna zona di disponibilità (AZ) deve avere almeno un indirizzo IP disponibile per ogni istanza di Neptune. Ad esempio, se ci sono un'istanza di scrittura e due istanze di lettura nella sottorete 1 e due istanze di lettura nella sottorete 2, la sottorete 1 deve avere almeno 3 indirizzi IP liberi e la sottorete 2 deve avere almeno 2 indirizzi IP liberi prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per alcuni minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> In questo rilascio è presente una modifica radicale per il codice che usa openCypher con autenticazione IAM. Finora, la stringa host nella firma IAM includeva il protocollo, come `bolt://`, ad esempio:
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> A partire dal rilascio del motore `1.1.1.0`, il protocollo deve essere omesso:
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> Per esempi, consulta Utilizzo del protocollo Bolt.

Miglioramenti in questo rilascio del motore

- Sono state migliorate le prestazioni delle query `order-by` Gremlin. Le query Gremlin con `order-by` al termine di un `NeptuneGraphQueryStep` ora utilizzano blocchi di dimensioni superiori per prestazioni migliori. Questo non si applica a `order-by` in un nodo interno (non root) del piano di query.
- Sono state migliorate le prestazioni delle query di aggiornamento Gremlin. I vertici e gli archi devono ora essere bloccati per impedirne l'eliminazione durante l'aggiunta di archi o proprietà. Questa modifica elimina i blocchi duplicati all'interno di una transazione, migliorando così le prestazioni.

Difetti corretti in questa versione del motore

- È stato corretto un bug di openCypher nella clausola `MERGE` che in alcuni casi causava la creazione di nodi e archi duplicati.
- È stato corretto un bug nella gestione delle query di SPARQL che contengono `(NOT) EXISTS` all'interno di una clausola `OPTIONAL`, a causa del quale in qualche caso mancavano alcuni risultati delle query.
- È stato corretto un bug che ritardava il riavvio del server quando era in corso un caricamento in blocco.

- È stato corretto un bug a causa del quale l'attraversamento bidirezionale di un modello a lunghezza variabile openCypher con un filtro sulla proprietà di relazione causava un errore. Un esempio di tale modello a lunghezza variabile è `(n)-[r*1..2]->(m)`.
- È stato corretto un bug relativo al modo in cui i dati memorizzati nella cache vengono inviati nuovamente al client, che in alcuni casi provocava una latenza inaspettatamente lunga.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.1.1.0.R6, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.1.1.0.R6

Se si esegue la versione del motore `1.1.1.0`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

Important

L'aggiornamento a questo rilascio del motore da qualsiasi versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di

5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per circa 6 minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di

inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.1.1.0.R5 (21/07/2022)

A partire dal 21 luglio 2022, viene implementata a livello generale la versione del motore 1.1.1.0.R5. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Important

L'aggiornamento a questo rilascio del motore da una versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
Per completare correttamente l'aggiornamento, ogni sottorete in ciascuna zona di disponibilità (AZ) deve avere almeno un indirizzo IP disponibile per ogni istanza di Neptune. Ad esempio, se ci sono un'istanza di scrittura e due istanze di lettura nella sottorete 1 e due istanze di lettura nella sottorete 2, la sottorete 1 deve avere almeno 3 indirizzi IP liberi e la sottorete 2 deve avere almeno 2 indirizzi IP liberi prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di

5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per alcuni minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

In questo rilascio è presente una modifica radicale per il codice che usa openCypher con autenticazione IAM. Finora, la stringa host nella firma IAM includeva il protocollo, come `bolt://`, ad esempio:

```
"Host":"bolt://(host URL):(port)"
```

A partire dal rilascio del motore `1.1.1.0`, il protocollo deve essere omesso:

```
"Host":"(host URL):(port)"
```

Per esempi, consulta Utilizzo del protocollo Bolt.

Miglioramenti in questo rilascio del motore

- Sono stati apportati miglioramenti per supportare il rilevamento dei deadlock.

Difetti corretti in questa versione del motore

- È stato corretto un bug che impediva l'arresto pulito dei cluster database in determinate condizioni.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.1.1.0.R5, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.1.1.0.R5

Se si esegue la versione del motore `1.1.1.0`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

Important

L'aggiornamento a questo rilascio del motore da qualsiasi versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del

sistema operativo. A questo punto, il cluster database non sarà disponibile per circa 6 minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di

inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.1.1.0.R4 (23/06/2022)

A partire dal 23 giugno 2022, viene implementata a livello generale la versione del motore 1.1.1.0.R4. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Important

L'aggiornamento a questo rilascio del motore da una versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.

Per completare correttamente l'aggiornamento, ogni sottorete in ciascuna zona di disponibilità (AZ) deve avere almeno un indirizzo IP disponibile per ogni istanza di Neptune. Ad esempio, se ci sono un'istanza di scrittura e due istanze di lettura nella sottorete 1 e due istanze di lettura nella sottorete 2, la sottorete 1 deve avere almeno 3 indirizzi IP liberi e la sottorete 2 deve avere almeno 2 indirizzi IP liberi prima di avviare l'aggiornamento.

All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.

La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di

5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per alcuni minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

In questo rilascio è presente una modifica radicale per il codice che usa openCypher con autenticazione IAM. Finora, la stringa host nella firma IAM includeva il protocollo, come `bolt://`, ad esempio:

```
"Host":"bolt://(host URL):(port)"
```

A partire dal rilascio del motore `1.1.1.0`, il protocollo deve essere omesso:

```
"Host":"(host URL):(port)"
```

Per esempi, consulta Utilizzo del protocollo Bolt.

Miglioramenti in questo rilascio del motore

- È stata aggiornata la configurazione dell'istanza per i tipi di istanza `x2g`.
- Sono state migliorate le prestazioni di rimozione dei vertici.

Difetti corretti in questa versione del motore

- È stato corretto un bug di Gremlin a causa del quale le soluzioni non mantenevano un ordine stabile per una query chiamata più volte o tra più istanze di lettura per determinati tipi di join ASK.
- Inoltre, è stato ristretto l'ambito di una modifica apportata nel rilascio precedente che causava regressioni delle prestazioni per alcuni tipi di join ASK in Gremlin.
- È stato corretto un bug di Gremlin nella fase `union()` che si verificava quando c'era un input di arco e un attraversamento verso un vertice all'interno degli attraversamenti figlio.
- È stato corretto un bug del profilo di Gremlin per cui alcune fasi venivano segnalate come non ottimizzate quando in realtà lo erano.
- È stato corretto un bug di SPARQL a causa del quale alle variabili utilizzate all'interno delle espressioni `FILTER` annidate nelle clausole `UNION` venivano assegnate informazioni di ambito non valide.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.1.1.0.R4, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.1.1.0.R4

Se si esegue la versione del motore `1.1.1.0`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

> **Important**
>
> L'aggiornamento a questo rilascio del motore da qualsiasi versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema

operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per circa 6 minuti.
Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale,

questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.1.1.0.R3 (07/06/2022)

A partire dal 7 giugno 2022, viene implementata a livello generale la versione del motore 1.1.1.0.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Important**
>
> L'aggiornamento a questo rilascio del motore da una versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
> All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni

del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento. La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per alcuni minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

In questo rilascio è presente una modifica radicale per il codice che usa openCypher con autenticazione IAM. Finora, la stringa host nella firma IAM includeva il protocollo, come `bolt://`, ad esempio:

```
"Host":"bolt://(host URL):(port)"
```

A partire dal rilascio del motore `1.1.1.0`, il protocollo deve essere omesso:

```
"Host":"(host URL):(port)"
```

Per esempi, consulta Utilizzo del protocollo Bolt.

Miglioramenti in questo rilascio del motore

- Aggiunto il supporto per i tipi di istanza `x2g` basati su Graviton2, ottimizzati per carichi di lavoro che richiedono molta memoria. Inizialmente sono disponibili solo in quattro Regioni AWS:
  - Stati Uniti orientali (Virginia settentrionale) (`us-east-1`)
  - Stati Uniti orientali (Ohio) (`us-east-2`)
  - Stati Uniti occidentali (Oregon) (`us-west-2`)
  - Europa (Irlanda) (`eu-west-1`)

  Per ulteriori informazioni, consulta la pagina dei prezzi di Neptune.
- Sono state migliorate le prestazioni delle fasi di Gremlin in cui sono coinvolti più attraversamenti di archi o vertici, ricerche di proprietà o ricerche di etichette.

Difetti corretti in questa versione del motore

- È stato corretto un bug di Gremlin nell'elaborazione della fase `otherV()` all'interno di un attraversamento figlio.
- È stato corretto un bug di Gremlin nelle query con `union` che hanno come elemento figlio solo fasi di filtro. Per esempio:

```
g.V().union(has("name"), out("knows")).out()
```

- È stato corretto un bug di SPARQL a causa del quale alle variabili utilizzate all'interno delle espressioni `FILTER` annidate nelle clausole `UNION` venivano assegnate informazioni di ambito non valide.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.1.1.0.R3, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.1.1.0.R3

Se si esegue la versione del motore `1.1.1.0`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

Important

L'aggiornamento a questo rilascio del motore da qualsiasi versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per circa 6 minuti.
Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Rilascio di manutenzione di Amazon Neptune, versione 1.1.1.0.R2 (16/05/2022)

A partire dal 16 maggio 2022, viene implementata a livello generale la versione del motore 1.1.1.0.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Important

L'aggiornamento a questo rilascio del motore da una versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
Per completare correttamente l'aggiornamento, ogni sottorete in ciascuna zona di disponibilità (AZ) deve avere almeno un indirizzo IP disponibile per ogni istanza di Neptune. Ad esempio, se ci sono un'istanza di scrittura e due istanze di lettura nella sottorete 1 e due istanze di lettura nella sottorete 2, la sottorete 1 deve avere almeno 3 indirizzi IP liberi e la sottorete 2 deve avere almeno 2 indirizzi IP liberi prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per alcuni minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:

- `Applying off-line patches to DB instance`
- `DB instance shutdown`
- `Finished applying off-line patches to DB instance`
- `DB instance restarted`

Note

In questo rilascio è presente una modifica radicale per il codice che usa openCypher con autenticazione IAM. Finora, la stringa host nella firma IAM includeva il protocollo, come `bolt://`, ad esempio:

```
"Host":"bolt://(host URL):(port)"
```

A partire dal rilascio del motore `1.1.1.0`, il protocollo deve essere omesso:

```
"Host":"(host URL):(port)"
```

Per esempi, consulta Utilizzo del protocollo Bolt.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.1.1.0.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione meno recente di Gremlin supportata: `3.5.2`
- Versione più recente di Gremlin supportata: `3.5.4`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.1.1.0.R2

Se si esegue la versione del motore `1.1.1.0`, il cluster viene aggiornato automaticamente a questo rilascio di patch di manutenzione durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Important

L'aggiornamento a questo rilascio del motore da qualsiasi versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per circa 6 minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.1.0.0 (19/11/2021)

A partire dal 19 novembre 2021, viene implementata a livello generale la versione del motore 1.1.0.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Important

L'aggiornamento a questo rilascio del motore da una versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
Per completare correttamente l'aggiornamento, ogni sottorete in ciascuna zona di disponibilità (AZ) deve avere almeno un indirizzo IP disponibile per ogni istanza di Neptune. Ad esempio, se ci sono un'istanza di scrittura e due istanze di lettura nella sottorete 1 e due istanze di lettura nella sottorete 2, la sottorete 1 deve avere almeno 3 indirizzi IP liberi e la sottorete 2 deve avere almeno 2 indirizzi IP liberi prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per alcuni minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`

- `DB instance shutdown`
- `Finished applying off-line patches to DB instance`
- `DB instance restarted`

> **Note**
>
> A partire da questo rilascio del motore, Neptune non supporta più le istanze di tipo R4. Se utilizzi un'istanza R4 nel cluster database, devi sostituirla manualmente con un'istanza di tipo diverso prima di eseguire l'aggiornamento a questo rilascio. Se l'istanza di scrittura è di tipo R4, segui queste istruzioni per spostarla.

Rilascio di patch successive per questo rilascio

- Rilascio di manutenzione: 1.1.0.0.R2 (16/05/2022)
- Rilascio di manutenzione: 1.1.0.0.R3 (23/12/2022)

Nuove caratteristiche in questa versione del motore

- Sono state introdotte istanze di database T4g per uso generale e R6g ottimizzate per la memoria basate sul processore Graviton2 AWS. Le istanze basate su Graviton2 offrono un rapporto prezzo/prestazioni significativamente migliore rispetto alle istanze basate su x86 di generazione attuale simili per una varietà di carichi di lavoro. Le applicazioni funzionano normalmente su questi nuovi tipi di istanze e non è necessario eseguire il porting del codice dell'applicazione quando si esegue l'aggiornamento ad esse.

  Per ulteriori informazioni sui prezzi e sulla disponibilità regionale, consulta la pagina relativa ai prezzi di Amazon Neptune.
- Sono stati introdotti alcuni modelli personalizzati in Neptune ML.
- Aggiunto il supporto per le query di inferenza SPARQL in Neptune ML.
- È stato aggiunto un nuovo endpoint streams per i dati dei grafi di proprietà, ossia:

```
https://Neptune-DNS:8182/propertygraph/stream
```

Il formato di output di questo endpoint, denominato `PG_JSON`, è esattamente lo stesso del formato di output `GREMLIN_JSON` del vecchio endpoint `gremlin/stream`.

Il nuovo endpoint `propertygraph/stream` estende il supporto per il flusso Neptune a openCypher e sostituisce l'endpoint `gremlin/stream` con il formato di output `GREMLIN_JSON` associato.

Miglioramenti in questo rilascio del motore

- Sono stati apportati miglioramenti ai flussi Neptune:
  - È stato aggiunto un campo `commitTimestamp` all'oggetto `records` nel formato di risposta del log delle modifiche dei flussi Neptune per fornire un timestamp per ogni record in un flusso di log delle modifiche.
  - È stato aggiunto un valore `LATEST` al parametro `iteratorType`, che consente di recuperare l'ultimo eventId valido dai flussi. Per informazioni, consultare Chiamata dell'API Streams.
- Aggiunto il supporto per ottenere il punteggio di attendibilità dell'inferenza nelle query di classificazione e regressione dei nodi Gremlin.
- Aggiunto il supporto per la clausola `OPTIONAL MATCH` in openCypher.
- Aggiunto il supporto per la clausola `MERGE` in openCypher.
- Aggiunto il supporto per l'utilizzo di `ORDER BY` nelle clausole `WITH` in openCypher.
- Aggiunto il supporto per la comprensione dei pattern in openCypher ed esteso il supporto per l'espressione dei pattern oltre il controllo dell'esistenza.
- È stato esteso il supporto per le clausole `DELETE DETACH` e `DELETE` in openCypher, in modo che possano ora essere utilizzate con altre clausole di aggiornamento.
- È stato esteso il supporto per le clausole `CREATE` e `UPDATE` utilizzate con `RETURN` in openCypher.
- È stato aggiunto il supporto nel motore DFE per le fasi `limit`, `range` e `skip` di Gremlin.
- È stata migliorata l'esecuzione di query nel motore DFE quando non è richiesto `explain` né `profile`.
- È stata migliorata l'esecuzione di query nel motore DFE per l'espressione `value`.
- Sono stati migliorati alcuni pattern di inserimento condizionale di Gremlin concatenati in modo da evitare eccezioni di modifica simultanea e da consentire il concatenamento di pattern di interrogazione come questi:
  - Inserimento condizionale di vertici per ID, ad esempio:

```
g.V(ID).fold().coalesce(unfold(), g.addV("L1").property(id,ID))
```

- Inserimento condizionale di vertici con più etichette, ad esempio:

```
g.V(ID).fold().coalesce(unfold(), g.addV("L1::L2").property(id,ID))
```

- Inserimento condizionale di archi per ID, ad esempio:

```
g.E(ID).fold().coalesce(unfold(), V(from).addE(label).to(V(to)).property(id, ID))
```

- Inserimento condizionale di archi con più etichette, ad esempio:

```
g.E(ID).fold().coalesce(unfold(),
 g.addE(label).from(V(from)).to(V(to)).property(id, ID))
```

- Inserimento condizionale seguito da una query, ad esempio:

```
g.V(ID).fold().coalesce(unfold(),
 g.addV("L1").property(id,ID)).project("myvalues").by(valueMap())
```

- Inserimento condizionale con proprietà aggiunte, ad esempio:

```
g.V(ID).fold().coalesce(unfold(),
 g.addV("L1").property(id,ID).property("name","pumba"))
```

Difetti corretti in questa versione del motore

- È stata disabilitata la funzionalità relativa alle statistiche sulle istanze di tipo `T3.medium`, che non erano in grado di supportarla.
- È stato corretto un bug SPARQL in `explain` relativo a una funzione `IN` che assumeva valori non costanti.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.1.0.0, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.11`

- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.1.0.0

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

L'aggiornamento a questo rilascio non viene eseguito automaticamente.

Aggiornamento a questo rilascio

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Invece di `--apply-immediately`, puoi specificare `--no-apply-immediately`. Per eseguire un aggiornamento di versione principale, è richiesto il parametro allow-major-version-upgrade. Assicurati inoltre di includere la versione del motore onde evitare che il tuo motore venga aggiornato a una versione diversa.

Se il cluster utilizza un gruppo di parametri del cluster personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Analogamente, se alcune istanze del cluster utilizzano un gruppo di parametri del database personalizzato, assicurati di includere questo parametro per specificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Rilascio di manutenzione di Amazon Neptune, versione 1.1.0.0.R3 (23/12/2022)

A partire dal 23 dicembre 2022, viene implementata a livello generale la versione del motore 1.1.0.0.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Important**
>
> L'aggiornamento a questo rilascio del motore da una versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
> Per completare correttamente l'aggiornamento, ogni sottorete in ciascuna zona di disponibilità (AZ) deve avere almeno un indirizzo IP disponibile per ogni istanza di Neptune. Ad esempio, se ci sono un'istanza di scrittura e due istanze di lettura nella sottorete 1 e due istanze di

lettura nella sottorete 2, la sottorete 1 deve avere almeno 3 indirizzi IP liberi e la sottorete 2 deve avere almeno 2 indirizzi IP liberi prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per alcuni minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Miglioramenti in questo rilascio del motore

- Miglioramenti delle prestazioni e correzioni della correttezza per vari operatori Gremlin, tra cui `repeat`, `coalesce`, `store` e `aggregate`.

Difetti corretti in questa versione del motore

- È stato risolto un problema relativo ai picchi della CPU.
- È stato corretto un bug di openCypher per cui le query restituivano la stringa `"null"` anziché un valore nullo in Bolt and SPARQL-JSON.

- È stato corretto un bug del log di audit che causava la registrazione di informazioni non necessarie e l'assenza di alcuni campi nei log.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.1.0.0.R3, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.11`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.1.0.0.R3

Se si esegue la versione del motore `1.1.0.0`, il cluster viene aggiornato automaticamente a questo rilascio di patch di manutenzione durante la finestra di manutenzione successiva.

> **Important**
>
> L'aggiornamento a questo rilascio del motore da qualsiasi versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
> All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
> La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per circa 6 minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
> Questo processo genera i seguenti eventi:
>
> - Messaggi di evento per cluster:

    - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
    - `Database cluster major version has been upgraded`
  - Messaggi di evento per istanza:
    - `Applying off-line patches to DB instance`
    - `DB instance shutdown`
    - `Finished applying off-line patches to DB instance`
    - `DB instance restarted`

> **Note**
>
> A partire da questo rilascio del motore, Neptune non supporta più le istanze di tipo R4. Se utilizzi un'istanza R4 nel cluster database, devi sostituirla manualmente con un'istanza di tipo diverso prima di eseguire l'aggiornamento a questo rilascio. Se l'istanza di scrittura è di tipo R4, segui queste istruzioni per spostarla.

Aggiornamento a questo rilascio

Amazon Neptune 1.1.0.0.R3 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Rilascio di manutenzione di Amazon Neptune, versione 1.1.0.0.R2 (16/05/2022)

A partire dal 16 maggio 2022, viene implementata a livello generale la versione del motore 1.1.0.0.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

> **Important**
>
> L'aggiornamento a questo rilascio del motore da una versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
> All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni

del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento. La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per alcuni minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Difetti corretti in questa versione del motore

- È stato corretto un bug a causa del quale la cache interna delle credenziali non veniva cancellata correttamente per gli endpoint non sottoposti a query, come l'endpoint status.
- È stato corretto un bug che causava un aumento del ritardo di replica dopo un aggiornamento del motore.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.1.0.0.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.11`
- Versione openCypher: `Neptune-9.0.20190305-1.0`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.1.0.0.R2

Se si esegue la versione del motore `1.1.0.0`, il cluster viene aggiornato automaticamente a questo rilascio di patch di manutenzione durante la finestra di manutenzione successiva.

Important

L'aggiornamento a questo rilascio del motore da qualsiasi versione precedente alla **`1.1.0.0`** attiva anche un aggiornamento del sistema operativo su tutte le istanze del cluster database. Poiché le richieste di scrittura attive che si verificano durante l'aggiornamento del sistema operativo non vengono elaborate, è necessario sospendere tutti i carichi di lavoro in scrittura sul cluster da aggiornare, compresi i caricamenti di dati in blocco, prima di avviare l'aggiornamento.
All'inizio dell'aggiornamento, Neptune genera uno snapshot con un nome composto da `preupgrade` seguito da un identificatore generato automaticamente in base alle informazioni del cluster database. Per lo snapshot non ti verrà addebitato alcun costo e potrai utilizzarlo per ripristinare il cluster database in caso di problemi durante il processo di aggiornamento.
La nuova versione del motore (una volta completato il relativo aggiornamento) rimane disponibile per un breve periodo di tempo sul vecchio sistema operativo, ma in meno di 5 minuti tutte le istanze del cluster avviano contemporaneamente un aggiornamento del sistema operativo. A questo punto, il cluster database non sarà disponibile per circa 6 minuti. Puoi riprendere i carichi di lavoro in scrittura al termine dell'aggiornamento.
Questo processo genera i seguenti eventi:

- Messaggi di evento per cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Messaggi di evento per istanza:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> A partire da questo rilascio del motore, Neptune non supporta più le istanze di tipo R4. Se utilizzi un'istanza R4 nel cluster database, devi sostituirla manualmente con un'istanza di tipo diverso prima di eseguire l'aggiornamento a questo rilascio. Se l'istanza di scrittura è di tipo R4, segui queste istruzioni per spostarla.

Aggiornamento a questo rilascio

Amazon Neptune 1.1.0.0.R2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un

aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.5.1 (01/10/2021)

A partire dal 1° ottobre 2021, viene implementata a livello generale la versione del motore 1.0.5.1. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Rilascio di patch successive per questa versione

- Rilascio: 1.0.5.1.R2 (26/10/2021)
- Rilascio: 1.0.5.1.R3 (13/01/2022.)
- Rilascio di manutenzione: 1.0.5.1.R4 (16/05/2022)

Nuove caratteristiche in questa versione del motore

- Aggiunta una cache dei risultati per memorizzare nella cache risultati delle query specificate.
- Aggiunto il supporto data/ora in Neptune openCypher.
- Aggiunto il supporto per l'accesso a `List` e `Map` di elementi in Neptune openCypher.

Miglioramenti in questo rilascio del motore

- I nomi degli endpoint di Neptune openCypher non fanno distinzione tra maiuscole e minuscole.
- Migliorata la spiegazione di openCypher.
- Miglioramento dei modelli di query di Gremlin a singolo upsert che terminano con i passaggi `iterate()` e `profile()`.
- Prestazioni migliorate in Gremlin e nelle sue funzioni `keys()` `property()`.

- Il passaggio `dedup()` di Gremlin viene eseguito nel DFE quando viene utilizzato con ambito globale.
- I seguenti predicati `HAS` di Gremlin vengono eseguiti nel motore DFE quando il motore DFE è abilitato:
  - `EQ`
  - `NEQ`
  - `LT`
  - `LTE`
  - `GT`
  - `GTE`
  - `BETWEEN`
  - `INSIDE`
  - `OUTSIDE`
  - `WITHIN`
  - `AND (connectives)`
  - `OR (connectives)`
- Prestazioni migliorate delle query LIMIT.
- Prestazioni migliorate delle query di aggregazione generali di openCypher.

Difetti corretti in questa versione del motore

- Corretto un bug di Gremlin che consentiva di collegare un edge a un altro edge.
- Corretto un bug di Gremlin che causava la scelta di una join strategy non ottimale.
- Corretto un bug di Gremlin che causava il blocco della serializzazione dei nodi e delle relazioni quando erano presenti più di 100 proprietà.
- Corretto un bug che rallentava la pianificazione dell'esecuzione delle query per le query con schemi grafici di grandi dimensioni.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.5.1, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.11`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.5.1

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Non +possibile effettuare l'aggiornamento automatico a questa versione.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.5.1 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
> ```

```
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Rilascio di manutenzione di Amazon Neptune, versione 1.0.5.1.R4 (16/05/2022)

A partire dal 16 maggio 2022, viene implementata a livello generale la versione del motore 1.0.5.1.R4. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.5.1.R4, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.11`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.5.1.R4

Se si esegue la versione del motore `1.0.5.1`, il cluster viene aggiornato automaticamente a questo rilascio di patch di manutenzione durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.5.1.R4 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.5.1.R3 (13/01/2022.)

A partire dal 13 gennaio 2022, viene implementata a livello generale la versione del motore 1.0.5.1.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Difetti corretti in questa versione del motore

- Corretto un bug che poteva causare una perdita di risorse quando una query non riusciva ad acquisire tutte le risorse necessarie.
- Corretta una piccola perdita di memoria durante l'esecuzione della query causata da un'allocazione di memoria non sfruttata.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.5.1.R3, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.11`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.5.1.R3

Se si esegue la versione del motore `1.0.5.1`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.5.1.R3 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.5.1.R2 (26/10/2021)

A partire dal 26 ottobre 2021, viene implementata a livello generale la versione del motore 1.0.5.1.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Difetti corretti in questa versione del motore

- Corretto un bug che causava il riavvio del server quando si verificava un errore transitorio durante la creazione di una versione precedente di un elemento grafico, con isolamento di lettura ripetibile. Ora invece Neptune restituisce un errore, in modo che il client possa riprovare.

- Corretto un bug che causava il riavvio del server quando si verificava un errore temporaneo durante un singolo aggiornamento della cardinalità. Ora invece Neptune restituisce un errore, in modo che il client possa riprovare.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.5.1.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.11`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.5.1.R2

Se si esegue la versione del motore `1.0.5.1`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.5.1.R2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.5.0 (27/07/2021)

A partire dal 27 luglio 2021, viene implementata a livello generale la versione del motore 1.0.5.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Rilascio di patch successive per questa versione

- Rilascio: 1.0.5.0.R2 (16/08/2021)
- Rilascio: 1.0.5.0.R3 (15/09/2021)
- Rilascio di manutenzione: 1.0.5.0.R5 (16/05/2022)

Nuove caratteristiche in questa versione del motore

- Neptune ML è stato rilasciato per l'uso in produzione con molte nuove funzionalità e non è più in modalità di laboratorio.

- Aggiunto il supporto iniziale per il linguaggio di query openCypher, in modalità di laboratorio. openCypher è lo standard open source per il linguaggio di query Cypher. La sua sintassi è specificata nel Cypher Query Language Reference (versione 9) ed è gestita dal progetto openCypher.

  Consulta Accesso al grafo di Neptune con openCypher per informazioni sull'implementazione del linguaggio Neptune.

  È supportato anche il supporto per il protocollo Bolt, utilizzato dai client Neptune per le query openCypher. Per informazioni, consultare Utilizzo del protocollo Bolt per effettuare query openCypher in Neptune.

  Il supporto per openCypher ora è abilitato automaticamente, ma dipende dal Motore DFE di Neptune, che attualmente è disponibile solo modalità di laboratorio. L'impostazione predefinita `DFEQueryEngine` nel parametro del cluster database `neptune_lab_mode` è ora `DFEQueryEngine=viaQueryHint`, il che significa che il motore è abilitato ma viene utilizzato solo per le query che hanno l'hint `useDFE` presente e impostato su `true`. Se disabiliti il motore DFE impostando `DFEQueryEngine=disabled`, non potrai utilizzare openCypher.
- Aggiunto il supporto per il protocollo HTTP SPARQL 1.1 Graph Store. Per informazioni, consultare Utilizzo del protocollo HTTP Graph Store Protocol (GSP) SPARQL 1.1 in Amazon Neptune.
- Modificata l'impostazione predefinita della modalità di laboratorio per il Motore DFE di Neptune a `viaQueryHint`, il che significa che il motore DFE è ora abilitato per impostazione predefinita, ma viene utilizzato solo per le query che hanno l'hint `useDFE` presente e impostato su `true`.
- Aggiunta una nuova metrica Amazon CloudWatch, `StatsNumStatementsScanned`, per monitorare il calcolo delle statistiche per il motore Neptune DFE. Per informazioni, consultare Utilizzo della `StatsNumStatementsScanned` CloudWatch metrica per monitorare il calcolo delle statistiche.

Miglioramenti in questo rilascio del motore

- Aggiunto il supporto per Apache TinkerPop 3.4.11.

> **Important**
>
> Nella versione 3.4.11 di TinkerPop è stata apportata una modifica che migliora la correttezza della modalità di elaborazione delle query, ma per il momento l'impatto sulle prestazioni delle query può essere significativo.

Una query di questo tipo, ad esempio, può essere eseguita molto più lentamente:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  out()
```

I vertici dopo il passaggio limite vengono ora recuperati in modo non ottimale a causa della modifica di TinkerPop 3.4.11. Per evitare il problema, puoi modificare la query aggiungendo il passaggio barrier() in qualsiasi punto dopo `order().by()`. Per esempio:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

- L'hint di query `joinOrder` di SPARQL è ora supportato dal motore di query alternativo Neptune DFE.
- L'output dell'API di stato Neptune è stato ampliato e riorganizzato per fornire maggiore chiarezza sulle impostazioni e le funzionalità del cluster database.

  Il nuovo output ha un oggetto `features` di primo livello che contiene informazioni sullo stato delle funzionalità del cluster database e un oggetto `settings` di primo livello che contiene informazioni sulle impostazioni. Per esaminare il nuovo formato, consulta Esempio di output del comando di stato dell'istanza.
- La gestione dei log delle modifiche allo streaming è stata migliorata quando gli stream `AFTER_SEQUENCE_NUMBER` vengono richiesti con l'ultimo ID evento sul server, quando tale ID evento è già scaduto. Il server non genera più un errore relativo all'ID evento scaduto se l'ID evento richiesto è l'ultimo ID evento eliminato sul server.

Difetti corretti in questa versione del motore

- Corretto un bug di Gremlin relativo all'ordinamento dei valori numerici.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.5.0, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.11`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.5.0

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Non +possibile effettuare l'aggiornamento automatico a questa versione.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.5.0 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di

inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Rilascio di manutenzione di Amazon Neptune, versione 1.0.5.0.R5 (16/05/2022)

A partire dal 16 maggio 2022, viene implementata a livello generale la versione del motore 1.0.5.0.R5. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.5.0.R5, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.11`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.5.0.R5

Se si esegue la versione del motore 1.0.5.0, il cluster verrà aggiornato automaticamente a questa patch release di manutenzione durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.5.0.R5 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.5.0.R3 (15/09/2021)

A partire dal 15 settembre 2021, viene implementata a livello generale la versione del motore 1.0.5.0.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Difetti corretti in questa versione del motore

- Corretto un bug che impediva al motore di rispondere in una di queste situazioni:
  - Un caricamento in blocco avviene contemporaneamente al calcolo automatico delle statistiche.
  - Un calcolo statistico è stato richiesto manualmente nello stesso momento in cui ce n'era uno già in corso.
- Corretto un bug nel rilevamento dei deadlock e nell'acquisizione dei blocchi che poteva causare il crash del motore.
- Corretto un bug di Gremlin a causa del quale il motore generava un errore quando rilevava dati sconosciuti da un endpoint ML remoto in una query di inferenza Gremlin.
- Corretti diversi bug nelle API di gestione dei modelli ML relativi ai processi di trasformazione dei modelli e ai suggerimenti sulle istanze.
- Corretto un bug che poteva causare il crash del motore durante la generazione di ID di nodi ed edge.
- Corretto un bug che rallentava la generazione di piani di query per le query con pattern grafici di grandi dimensioni.
- Corretto un bug di openCypher che poteva causare il blocco di una query durante il recupero di un nodo con più di 100 proprietà.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.5.0.R3, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.11`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.5.0.R3

Se si esegue la versione del motore 1.0.5.0, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.5.0.R3 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un

aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.5.0.R2 (16/08/2021)

A partire dal 16 agosto 2021, viene implementata a livello generale la versione del motore 1.0.5.0.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Difetti corretti in questa versione del motore

- Disattivata un'ottimizzazione effettuata nel rilascio del motore `1.0.5.0` che consente di non svuotare la cache di ricerca di Neptune al riavvio del motore sulle repliche. Ora il riavvio della replica cancella la cache di ricerca.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.5.0.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.11`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.5.0.R2

Se si esegue la versione del motore `1.0.5.0`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.5.0.R2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.4.2 (01/06/2021)

> **Note**
>
> La versione di rilascio del motore 1.0.4.2.R2 è stata la prima versione di 1.0.4.2 effettivamente rilasciata.

Argomenti

Motore Amazon Neptune versione 1.0.4.2.R5 (16/08/2021)

A partire dal 16 agosto 2021, viene implementata a livello generale la versione del motore 1.0.4.2.R5. Tieni presente che occorrono diversi giorni prima che un nuovo rilascio diventi disponibile in ogni regione.

Difetti corretti in questa versione del motore

- Disattivata un'ottimizzazione effettuata nel rilascio del motore `1.0.4.2.R4` che consente di non svuotare la cache di ricerca di Neptune al riavvio del motore sulle repliche. Ora il riavvio della replica cancella la cache di ricerca.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.4.2.R5, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.10`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.4.2.R5

Se si esegue la versione del motore `1.0.4.2`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Motore Amazon Neptune versione 1.0.4.2.R4 (23/07/2021)

A partire dal 23 luglio 2021, viene implementata a livello generale la versione del motore 1.0.4.2.R4. Tieni presente che occorrono diversi giorni prima che un nuovo rilascio diventi disponibile in ogni regione.

Miglioramenti in questo rilascio del motore

- Migliorato il comportamento della cache di ricerca per evitare la cancellazione ridondante della cache dopo aver eseguito il ripristino rapido su una replica.
- Migliore gestione dei log delle modifiche in streaming quando vengono richiesti flussi `AFTER_SEQUENCE_NUMBER` con l'ultimo ID evento sul server, quando tale ID evento è già scaduto. Il server non genera più un errore relativo all'ID evento scaduto se l'ID evento richiesto è l'ultimo ID evento eliminato sul server.

Difetti corretti in questa versione del motore

- Corretto un bug introdotto in 1.0.4.0.R1 per cui le query non restituivano la totalità dei valori di stringa maggiori di 760 caratteri. I termini interessati da questo bug erano valori letterali e URI RDF o ID di Gremlin, chiavi e valori di stringa.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.4.2.R4, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.10`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.4.2.R4

Se si esegue la versione del motore `1.0.4.2`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Motore Amazon Neptune versione 1.0.4.2.R3 (28/06/2021)

A partire dal 28 giugno 2021, viene implementata a livello generale la versione del motore 1.0.4.2.R3. Tieni presente che occorrono diversi giorni prima che un nuovo rilascio diventi disponibile in ogni regione.

Problemi noti in questo rilascio del motore

Problema:

Un bug di SPARQL che non rispetta il tipo di supporto in un'intestazione `Accept` se sono presenti spazi.

Ad esempio, una query con `-H "Accept: text/csv; q=1.0, */*; q=0.1"` restituisce un output JSON anziché un output CSV.

Soluzione alternativa:

Se rimuovi gli spazi nella clausola `Accept` nell'intestazione, il motore restituisce l'output nel corretto formato richiesto. In altre parole, anziché `-H "Accept: text/csv; q=1.0, */*; q=0.1"` , usa:

```
-H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

Difetti corretti in questa versione del motore

- Corretto un bug nella cancellazione della cache di ricerca sulle repliche dopo un ripristino rapido.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.4.2.R3, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.10`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.4.2.R3

Questo rilascio di patch è facoltativo a meno che il cluster database non utilizzi una o più istanze `R5d`. Se il cluster dispone di istanze `R5d`, verrà aggiornato automaticamente nella successiva finestra di manutenzione. Altrimenti, non verrà aggiornato automaticamente a questo rilascio di patch.

È possibile aggiornare manualmente il rilascio `1.0.4.2.R2` a questo rilascio`1.0.4.2.R3` utilizzando il comando AWS CLI apply-pending-maintenance-action (APIApplyPendingMaintenanceAction).

Motore Amazon Neptune versione 1.0.4.2.R2 (01/06/2021)

A partire dal 1° giugno 2021, viene implementata a livello generale la versione del motore 1.0.4.2.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Rilascio di patch successive per questa versione

- Rilascio: 1.0.4.2.R3 (28/06/2021)

Problemi noti in questo rilascio del motore

Problema:

Un bug di SPARQL che non rispetta il tipo di supporto in un'intestazione `Accept` se sono presenti spazi.

Ad esempio, una query con `-H "Accept: text/csv; q=1.0, */*; q=0.1"` restituisce un output JSON anziché un output CSV.

Soluzione alternativa:

Se rimuovi gli spazi nella clausola `Accept` nell'intestazione, il motore restituisce l'output nel corretto formato richiesto. In altre parole, anziché `-H "Accept: text/csv; q=1.0, */*; q=0.1"` , usa:

```
  -H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

Nuove caratteristiche in questa versione del motore

- Aggiunto il nuovo tipo di istanza R5d, che include una cache di ricerca per velocizzare le letture nei casi d'uso che coinvolgono un volume elevato di valori di proprietà o ricerche letterali RDF. Per informazioni, consultare La cache di ricerca di Neptune può accelerare le query di lettura.
- Aggiunto un nuovo parametro lab-mode che consente di richiamare il motore DFE sperimentale solo per ogni query con l'hint `useDFE`.

Miglioramenti in questo rilascio del motore

- Aggiunto il supporto per TinkerPop 3.4.10.
- Aggiunto il supporto per l'utilizzo del passaggio di configurazione `withStrategies( )` durante l'invio di richieste di script di Gremlin. Nello specifico, `SubgraphStrategy`, `PartitionStrategy`, `ReadOnlyStrategy`, `EdgeLabelVerificationStrategy`, e `ReservedKeysVerificationStrategy` sono tutti supportati.
- Aggiunta l'ottimizzazione per gli attraversamenti `V()` durante una query. In precedenza, tali attraversamenti non erano ottimizzati su Neptune.
- Aggiunto il supporto per gli URN RFC 2141 da utilizzare come parametri `baseUri` e `namedGraphUri` per un caricamento in blocco.

Difetti corretti in questa versione del motore

- Corretto un bug di Gremlin nel parser in cui le query errate venivano considerate valide.
- Corretto un bug di Gremlin per cui la visualizzazione di un effetto collaterale `aggregate()` con `cap().unfold()` a un `valueMap()` generava un'eccezione.
- Corretto un bug di Gremlin a causa del quale alcuni passaggi `property()` dopo un passaggio `addV()` non riuscivano con l'errore "impossibile eseguire il cast su String".
- Corretto un bug di Gremlin che impediva ad alcuni schemi di inserimento condizionali di generare eccezioni di modifica simultanea.
- Corretto un bug di Gremlin in modo che il timeout della richiesta di query ora non possa superare il timeout della sessione.

- Corretto un bug di SPARQL a causa del quale gli aggiornamenti tramite LOAD o UNLOAD potevano non riuscire con un codice HTTP 500 anziché il codice HTTP 400 quando il server remoto non era disponibile.
- Corretto un bug a causa del quale le chiamate API di streaming non riuscivano quando venivano utilizzati valori `commitNum` o valori `opNum` superiori al limite di numeri interi segnati a 32 bit (2.147.483.647).

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.4.2.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.10`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.4.2.R2

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Non +possibile effettuare l'aggiornamento automatico a questa versione.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.4.2.R2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.2 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.2 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.4.2.R1 (27/05/2021)

Il rilascio 1.0.4.2.R1 del motore non è mai stato implementato.

Motore Amazon Neptune versione 1.0.4.1 (08/12/2020)

A partire dall'08 dicembre 2020, viene implementata a livello generale la versione del motore 1.0.4.1. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Rilascio di patch successive per questa versione

- Rilascio: 1.0.4.1.R1.1 (22/03/2021)
- Rilascio: 1.0.4.1.R2 (24/02/2021)

> **Important**
>
> Rilascio: 1.0.4.0 (12/10/2020) ha reso TLS 1.2 e HTTPS obbligatori per tutte le connessioni ad Amazon Neptune. Tuttavia, un bug presente in quel rilascio ha consentito alle

connessioni HTTP e/o alle connessioni TLS obsolete di continuare a funzionare per i clienti che in precedenza avevano impostato un parametro del cluster database per impedire l'imposizione di connessioni HTTPS.
Questo bug è stato corretto nei rilasci di patch 1.0.4.0.R2 e 1.0.4.1.R2, ma la correzione ha causato errori di connessione imprevisti quando le patch vengono installate automaticamente. Per questo motivo, entrambe le patch sono state ripristinate e possono essere installate solo manualmente, per darti la possibilità di aggiornare la configurazione per TLS 1.2.
L'uso di SSL/TLS per tutte le connessioni a Neptune influisce sulle connessioni con la console di Gremlin, il driver di Gremlin, Gremlin Python, .NET, Node.js, le REST API e anche le connessioni del sistema di bilanciamento del carico. Se fino ad ora hai utilizzato HTTP o una versione TLS precedente per alcune o tutte queste versioni, devi aggiornare il client e i driver pertinenti e modificare il codice per utilizzare esclusivamente HTTPS prima di aggiornare il sistema alle patch più recenti.

Nuove caratteristiche in questa versione del motore

- È stata introdotta la funzionalità Neptune ML, che offre potenti funzionalità di machine learning ad Amazon Neptune. Per informazioni, consultare Amazon Neptune ML per machine learning sui grafi.
- È stata aggiunta un'operazione `UNLOAD` di SPARQL personalizzata per rimuovere i dati recuperati da una fonte remota. Per informazioni, consultare SPARQL UPDATE UNLOAD.

Miglioramenti in questo rilascio del motore

- Sono stati ottimizzati alcuni modelli di inserimento condizionale di Gremlin per evitare eccezioni di modifica simultanea.

Difetti corretti in questa versione del motore

- Corretto un bug di Gremlin che poteva causare la mancanza di risultati per uno specifico modello di query che utilizzava il passaggio `as()`.
- Corretto un bug di Gremlin che poteva causare errori quando si utilizzava il passaggio `project()` nidificato all'interno di un altro passaggio, ad esempio `union()`.
- Corretto un bug di Gremlin nel passaggio `project()`.

- Corretto un bug di Gremlin basato su attraversamenti di stringhe per cui il passaggio `none()` non funziona.
- Corretto un bug di Gremlin basato su attraversamenti di stringhe per cui una mappa vuota non viene supportata come argomentativo del passaggio `inject()`.
- Corretto bug di Gremlin basato sull'esecuzione di attraversamenti di stringhe nel motore DFE, a causa del quale un metodo terminale come `toList()` non funzionava correttamente.
- Corretto un bug di Gremlin che impediva la chiusura delle transazioni che utilizzavano il passaggio `iterate()` nella query String.
- Corretto un bug di Gremlin che poteva far sì che le query che utilizzavano il pattern `is(P.gte(0))` generassero un'eccezione in alcune situazioni.
- Corretto un bug di Gremlin che poteva far sì che le query che utilizzavano il pattern `order().by(T.id)` generassero un'eccezione in alcune situazioni.
- Corretto un bug di Gremlin che poteva far sì che le query che utilizzavano il pattern `addV().aggregate()` fornissero risultati errati in alcune situazioni.
- Corretto un bug di Gremlin che poteva far sì che le query che utilizzavano il passaggio `path()` seguito dallo schema del passaggio `project()` generassero un'eccezione in alcune situazioni.
- Corretto un bug di SPARQL in cui la funzione `SUBSTR` segnala un errore anziché restituire una stringa vuota.
- Corretto un bug nel motore DFE che poteva far sì che le operazioni di join nei piani di query non bloccanti generassero risultati errati in presenza di variabili non associate.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.4.1, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.8`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.4.1

Se si esegue la versione del motore `1.0.4.1`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.4.1 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.4.1.R1.1 (22/03/2021)

A partire dal 22 marzo 2021, viene implementata a livello generale la versione del motore 1.0.4.1.R1.1. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Difetti corretti in questa versione del motore

- Disattivata un'ottimizzazione per i pattern di inserimento condizionali di Gremlin che possono essere aggiunti o integrati a etichette e proprietà esistenti.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.4.1.R1.1, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.8`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.4.1.R1.1

Se si esegue la versione del motore `1.0.4.1`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.4.1.R1.1 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale,

questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.4.1.R2 (24/02/2021)

A partire dal 24 febbraio 2021, viene implementata a livello generale la versione del motore 1.0.4.1.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Rilascio di patch successive per questa versione

- Rilascio: 1.0.4.1.R2.1 (11/03/2021)

Nuove caratteristiche in questa versione del motore

- Ora Neptune supporta la compressione di file singoli in formato `bzip2` per il caricamento in blocco. Per informazioni, consultare Formati dei dati di caricamento.

Difetti corretti in questa versione del motore

- Corretto un bug su Rilascio: 1.0.4.0 (12/10/2020) che consentiva le connessioni a Neptune utilizzando HTTP o versioni precedenti di TLS, anziché HTTPS TLS 1.2.

  > **Important**
  >
  > La necessità di utilizzare SSL/TLS per tutte le connessioni a Neptune può costituire un cambiamento radicale. Influisce sulle connessioni con la console di Gremlin, il driver di Gremlin, Gremlin Python, .NET, Node.js, le REST API e anche le connessioni del sistema di bilanciamento del carico. Se finora hai utilizzato HTTP o una versione TLS precedente per alcune o tutte queste versioni, devi aggiornare il client e i driver pertinenti prima di installare questa patch e modificare il codice per utilizzare esclusivamente HTTPS.

- Corretto un bug di Gremlin in cui `InternalFailureException` veniva impostata come codice di risposta in determinate circostanze quando si verificava un'`ConcurrentModificationException`.
- Corretto un bug di Gremlin che, in determinate condizioni, aggiornando edge o vertici poteva causare un'`InternalFailureException` transitoria.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.4.1.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.8`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.4.1.R2

Se si esegue la versione del motore `1.0.4.1`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.4.1.R2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.4.1.R2.1 (11/03/2021)

A partire dall'11 marzo 2021, viene implementata a livello generale la versione del motore 1.0.4.1.R2.1. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Difetti corretti in questa versione del motore

- Disattivata un'ottimizzazione per i pattern di inserimento condizionali di Gremlin che possono essere aggiunti o integrati a etichette e proprietà esistenti.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.4.1.R2.1, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.8`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.4.1.R2.1

Se si esegue la versione del motore `1.0.4.1.R2`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.4.1.R2.1 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1.R2 \
```

```
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1.R2 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come

lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.4.0 (12/10/2020)

A partire dal 12 ottobre 2020, viene implementata a livello generale la versione del motore 1.0.4.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Rilascio di patch successive per questa versione

- Rilascio: 1.0.4.0.R2 (24/02/2021)

Nuove caratteristiche in questa versione del motore

- Aggiunta la compressione a livello di frame per Gremlin.

Miglioramenti in questo rilascio del motore

- Amazon Neptune ora richiede l'uso del Secure Sockets Layer (SSL) con il protocollo TLSv1.2 per tutte le connessioni a Neptune in tutte le regioni, utilizzando queste suite di crittografia avanzata:
  - `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
  - `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
  - `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
  - `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
  - `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
  - `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

  Questo vale sia per le connessioni REST che WebSocket a Neptune e significa che è necessario utilizzare HTTPS anziché HTTP per la connessione a Neptune in tutte le regioni.

  Poiché le connessioni client che utilizzano HTTP o TLS 1.1 non saranno più supportate da nessuna parte, assicurati che i client e il codice siano stati aggiornati per utilizzare TLS 1.2 e HTTPS prima di eseguire l'aggiornamento a questo rilascio del motore.

> **Important**
>
> La necessità di utilizzare SSL/TLS per tutte le connessioni a Neptune può costituire un cambiamento radicale. Influisce sulle connessioni con la console di Gremlin, il driver di Gremlin, Gremlin Python, .NET, Node.js, le REST API e anche le connessioni del sistema di bilanciamento del carico. Se hai utilizzato HTTP per alcune o tutte queste opzioni, ora devi aggiornare il client e i driver pertinenti, oltre che modificare il codice per utilizzare HTTPS, altrimenti le connessioni non riescono.
> Un bug in questo rilascio ha consentito alle connessioni HTTP e/o alle connessioni TLS obsolete di continuare a funzionare per i clienti che in precedenza avevano impostato un parametro del cluster database per impedire l'applicazione delle connessioni HTTPS. Questo bug è stato corretto nei rilasci di patch 1.0.4.0.R2 e 1.0.4.1.R2, ma la correzione ha causato errori di connessione imprevisti quando le patch vengono installate automaticamente.
> Per questo motivo, entrambe le patch sono state ripristinate e possono essere installate solo manualmente, per darti la possibilità di aggiornare la configurazione per TLS 1.2.

- Aggiornamento a TinkerPop versione 3.4.8. Si tratta di un aggiornamento retrocompatibile. Consulta il log delle modifiche di TinkerPop per sapere le novità.
- Miglioramento delle prestazioni per il passaggio `properties()` di Gremlin.
- Aggiunti dettagli su `BindOp` e `MultiplexerOp` nei rapporti esplicativi e di profilo.
- Aggiunto il prefetch dei dati per migliorare le prestazioni in caso di errori nella cache.
- Aggiunta una nuova impostazione `allowEmptyStrings` nel parametro `parserConfiguration` dello strumento di caricamento in blocco che consente di trattare le stringhe vuote come valori di proprietà validi nei caricamenti in formato CSV (consulta Parametri della richiesta dello dello strumento di caricamento Neptune).
- Lo strumento di caricamento consente ora l'escape del punto e virgola nelle colonne CSV a più valori.

Difetti corretti in questa versione del motore

- Corretta una potenziale perdita di memoria di Gremlin relativa al passaggio `both()`.
- Corretto un bug in cui mancavano le metriche di richiesta perché un endpoint che terminava con '/' non veniva gestito correttamente.
- Corretto un bug che causava il ritardo delle repliche e il riavvio in condizioni di carico elevato quando il motore DFE era abilitato in modalità di laboratorio.
- Corretto un bug che impediva la segnalazione del messaggio di errore corretto quando un caricamento in blocco non riusciva a causa di una condizione di esaurimento della memoria.
- Corretto un bug di SPARQL in cui la codifica dei caratteri veniva inserita nell'intestazione Content-Encoding nelle risposte alle query SPARQL. Ora `charset` viene invece inserito nell'intestazione Content-Type, per consentire ai client HTTP di riconoscere automaticamente il set di caratteri utilizzato.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.4.0, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.8`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.4.0

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Non +possibile effettuare l'aggiornamento automatico a questa versione.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.4.0 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.4.0.R2 (24/02/2021)

A partire dal 24 febbraio 2021, viene implementata a livello generale la versione del motore 1.0.4.0.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Difetti corretti in questa versione del motore

- Corretto un bug su Rilascio: 1.0.4.0 (12/10/2020) che consentiva le connessioni a Neptune utilizzando HTTP o versioni precedenti di TLS, anziché HTTPS TLS 1.2.

  > **Important**
  >
  > La necessità di utilizzare SSL/TLS per tutte le connessioni a Neptune può costituire un cambiamento radicale. Influisce sulle connessioni con la console di Gremlin, il driver di Gremlin, Gremlin Python, .NET, Node.js, le REST API e anche le connessioni del sistema di bilanciamento del carico. Se finora hai utilizzato HTTP o una versione TLS precedente per alcune o tutte queste versioni, devi aggiornare il client e i driver pertinenti prima di installare questa patch e modificare il codice per utilizzare esclusivamente HTTPS.

- Corretto un bug nel caricamento in blocco del file CSV che riguardava le etichette che terminavano con #.
- Corretto un bug di Gremlin in cui `InternalFailureException` veniva impostata come codice di risposta in determinate circostanze quando si verificava un'`ConcurrentModificationException`.
- Corretto un bug di Gremlin che, in determinate condizioni, aggiornando edge o vertici poteva causare un'`InternalFailureException` transitoria.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.4.0.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.8`

- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.4.0.R2

Se si esegue la versione del motore `1.0.4.0`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.4.0.R2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.3.0 (03/08/2020)

A partire dal 03 agosto 2020, viene implementata a livello generale la versione del motore 1.0.3.0. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Rilascio di patch successive per questa versione

- Rilascio: 1.0.3.0.R2 (12/10/2020)
- Rilascio: 1.0.3.0.R3 (19/02/2021)

Nuove caratteristiche in questa versione del motore

- Neptune ha introdotto un nuovo motore di query alternativo (DFE) che può velocizzare notevolmente l'esecuzione delle query. Per informazioni, consultare Il motore di query alternativo (DFE) di Amazon Neptune.
- Il DFE si basa su statistiche pregenerate sui dati a grafo di Neptune che vengono gestite tramite nuovi endpoint statistici. Per informazioni, consultare Statistiche DFE.
- È ora possibile escludere i processi di caricamento in coda dall'elenco degli ID di carico restituiti dall'API Loader Get-Status impostando il nuovo parametro `includeQueuedLoads` su FALSE. Per informazioni, consultare Parametri della richiesta Get-Status dello strumento di caricamento Neptune.
- Ora Neptune supporta le intestazioni finali per le risposte alle query SPARQL che possono contenere un codice di errore e un messaggio se una richiesta non riesce dopo che inizia a restituire blocchi di risposta. Per informazioni, consultare Intestazioni HTTP finali opzionali per risposte SPARQL in più parti.

- Ora Neptune consente anche di abilitare la codifica delle risposte in blocchi per le query di Gremlin. Come nel caso di SPARQL, i blocchi di risposta hanno delle intestazioni finali che possono contenere un codice di errore e un messaggio se si verifica un errore dopo che la query ha iniziato a restituire blocchi di risposta. Per informazioni, consultare Utilizzo di intestazioni HTTP finali opzionali per abilitare le risposte Gremlin in più parti.

Miglioramenti in questo rilascio del motore

- Ora puoi fornire la dimensione delle richieste batch a ElasticSearch per le ricerche full-text in Gremlin.
- Utilizzo ottimizzato della memoria per le query query GROUP BY di SPARQL.
- Aggiunto un nuovo ottimizzatore di query di Gremlin per eliminare alcuni filtri non associati.
- Aumentato il tempo massimo per cui una connessione WebSocket autenticata tramite IAM può rimanere aperta, da 36 ore a 10 giorni.

Difetti corretti in questa versione del motore

- Corretto un bug per cui se si inviava un parametro URL non codificato in una richiesta POST, Neptune restituiva un codice di stato HTTP di 500 e un'`InternalServerErrorException`. Ora Neptune restituisce un codice di stato HTTP 400 e un'`BadRequestException`, con il messaggio: `Failure to process the POST request parameters`.
- Corretto un bug di Gremlin in cui un errore di connessione WebSocket non veniva segnalato correttamente.
- Corretto un bug di Gremlin che comportava la scomparsa di sideEffect.
- Corretto un bug di Gremlin a causa del quale il parametro di ricerca full-text `batchsize` non era supportato correttamente.
- Corretto un bug di Gremlin da gestire `toV` e `fromV` singolarmente per ogni direzione su `bothE`.
- Corretto un bug di Gremlin che riguardava `Edge pathType` nel passaggio `hasLabel`.
- Corretto un bug di SPARQL a causa del quale il riordino dei join con associazioni statiche non funzionava correttamente.
- Corretto un bug di SPARQL UPDATE LOAD a causa del quale un bucket Amazon S3 non disponibile non veniva segnalato correttamente.
- Corretto un bug di SPARQL a causa del quale un problema con un nodo SERVICE in una sotto query non veniva segnalato correttamente.

- Corretto un bug di SPARQL a causa del quale le query contenenti condizioni FILTER EXISTS o FILTER NOT EXISTS nidificate non venivano valutate correttamente.
- Corretto un bug di SPARQL per gestire correttamente le associazioni duplicate generate quando si chiamano gli endpoint del servizio SPARQL attraverso la generazione di query.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.3.0, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.3`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.3.0

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Se il cluster ha il parametro `AutoMinorVersionUpgrade` impostato su `True`, il cluster verrà aggiornato automaticamente a questo rilascio del motore due o tre settimane dopo la data di questa versione, nel corso di una finestra di manutenzione.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.3.0 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale,

questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.3.0.R3 (19/02/2021)

A partire dal 19 febbraio 2021, viene implementata a livello generale la versione del motore 1.0.3.0.R3. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Difetti corretti in questa versione del motore

- Corretto un bug nel caricamento in blocco del file CSV che riguardava le etichette che terminavano con #.
- Corretto un bug di Gremlin che poteva causare la mancanza di risultati per un pattern specifico di query che utilizzavano il passaggio `as()`.
- Corretto un bug di Gremlin che poteva causare errori quando si utilizzava il passaggio `project()` nidificato all'interno di un altro passaggio, ad esempio `union()`.
- Corretto un bug di Gremlin nell'esecuzione di attraversamenti di stringhe nel motore sperimentale DFE quando si utilizza un metodo terminale simile a `toList()`.

- Corretto un bug di Gremlin che non riusciva a chiudere una transazione quando si utilizza il passaggio `iterate()` in una query di stringa.
- Corretto un bug di Gremlin che poteva far sì che le interrogazioni che utilizzavano il pattern `is(P.gte(0))` generassero un'eccezione in determinate condizioni.
- Corretto un bug di Gremlin in cui `InternalFailureException` veniva impostata come codice di risposta in determinate circostanze quando si verificava un'`ConcurrentModificationException`.
- Corretto un bug di Gremlin che, in determinate condizioni, aggiornando edge o vertici poteva causare un'`InternalFailureException` transitoria.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.3.0.R3, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.8`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.3.0.R3

Se si esegue la versione del motore `1.0.3.0`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.3.0.R3 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
```

```
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.3.0.R2 (12/10/2020)

A partire dal 12 ottobre 2020, viene implementata a livello generale la versione del motore 1.0.3.0.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Miglioramenti in questo rilascio del motore

- Miglioramento delle prestazioni per il passaggio `properties()` di Gremlin.
- Aggiunti dettagli su `BindOp` e `MultiplexerOp` nei rapporti esplicativi e di profilo.
- Per le risposte alle query SPARQL, aggiunto `charset` all'intestazione Content-Type, per consentire ai client HTTP di riconoscere automaticamente il set di caratteri utilizzato.

Difetti corretti in questa versione del motore

- Corretto un bug di SPARQL a causa del quale `CancellationException` non veniva gestita.
- Corretto un bug di SPARQL a causa del quale le query contenenti opzioni nidificate non funzionavano correttamente.
- Corretto un bug di SPARQL in LOAD in cui un'`ConcurrentModificationException` poteva causare il blocco di una query.
- Corretto un bug di SPARQL che impediva di comprimere le risposte alle query con gzip.
- Corretto un bug di Gremlin nel passaggio `groupBy()`.
- Corretto un bug di Gremlin relativo all'uso di un passaggio `aggregate()` all'interno di un passaggio `local()`.
- Corretto un bug di Gremlin relativo all'utilizzo di `bothE()` seguito da un predicato che utilizza valori aggregati.
- Corretto un bug di Gremlin relativo all'utilizzo del passaggio `bothE()` con il passaggio `repeat()`.
- Corretta una potenziale perdita di memoria di Gremlin relativa al passaggio `both()`.
- Corretto un bug in cui mancavano le metriche di richiesta perché un endpoint che terminava con '/' non veniva gestito correttamente.
- Corretto un bug che poteva generare un'`ThrottlingException` anche quando la coda delle richieste non era piena.
- Corretto un bug nel recupero dello stato del caricamento quando un caricamento non riesce per un motivo come `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE`.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.3.0.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.3`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.3.0.R2

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Se il cluster ha il parametro `AutoMinorVersionUpgrade` impostato su `True`, il cluster verrà aggiornato automaticamente a questo rilascio del motore due o tre settimane dopo la data di questa versione, nel corso di una finestra di manutenzione.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.3.0.R2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.2.2 (09/03/2020)

A partire dal 9 marzo 2020, viene implementata a livello generale la versione del motore 1.0.2.2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Rilascio di patch successive per questa versione

- Rilascio: 1.0.2.2.R2 (02/04/2020)
- Rilascio: 1.0.2.2.R3 (22/07/2020)
- Rilascio: 1.0.2.2.R4 (23/07/2020)
- Rilascio: 1.0.2.2.R5 (12/10/2020)
- Rilascio: 1.0.2.2.R6 (19/02/2021)

Miglioramenti in questo rilascio del motore

- Aggiunte informazioni all'API di stato sulle transazioni che vengono ripristinate. Per informazioni, consultare Stato dell'istanza.
- Aggiornato la versione di Apache TinkerPop alla 3.4.3.

  La versione 3.4.3 è retrocompatibile con la versione precedente supportata da Neptune (3.4.1). Introduce una modifica minore nel comportamento: Gremlin non restituisce più un errore quando si tenta di chiudere una sessione che non esiste (vedere Impedire l'insorgere di un errore quando si chiudono sessioni che non esistono).
- Rimossi i colli di bottiglia delle prestazioni durante l'esecuzione dei passaggi di ricerca full-text di Gremlin.

Difetti corretti in questa versione del motore

- Corretto un bug di SPARQL nella gestione di schemi grafici vuoti nelle query.

- Corretto un bug di SPARQL nella gestione di punti e virgola non codificati nelle query con codifica URL.
- Corretto un bug di Gremlin nella gestione dei vertici ripetuti nel passaggio `Union`.
- Corretto un bug Gremlin che portava alcune query con un `.simplePath()` o `.cyclicPath()` all'interno di un `.repeat()` a restituire risultati errati.
- Corretto un bug di Gremlin che portava `.project()` a restituire risultati errati se il suo attraversamento figlio non restituiva alcuna soluzione.
- Corretto un bug di Gremlin in cui gli errori da conflitti di lettura-scrittura sollevavano `InternalFailureException` piuttosto che `ConcurrentModificationException`.
- Corretto un bug di Gremlin che causava errori `.group().by(...).by(values("property"))`.
- Corretti i bug di Gremlin nell'output del profilo per i passaggi di ricerca full-text.
- Corretta una perdita di risorse nelle sessioni di Gremlin.
- Corretto un bug che impediva all'API di stato di segnalare la versione ordinabile corretta in alcuni casi.
- Corretto un bug dello strumento di caricamento in blocco che consentiva di utilizzare un URL a una posizione diversa da Amazon S3 come origine in una richiesta di caricamento in blocco.
- Corretto un bug dello strumento di caricamento in blocco nello stato di caricamento dettagliato.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.2.2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.3`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.2.2

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Se il cluster ha il parametro `AutoMinorVersionUpgrade` impostato su `True`, il cluster verrà aggiornato automaticamente a questo rilascio del motore due o tre settimane dopo la data di questa versione, nel corso di una finestra di manutenzione.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.2.2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.2.2.R6 (19/02/2021)

A partire dal 19 febbraio 2021, viene implementata a livello generale la versione del motore 1.0.2.2.R6. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Difetti corretti in questa versione del motore

- Corretto un bug di Gremlin in cui `InternalFailureException` veniva impostata come codice di risposta in determinate circostanze quando si verificava un'`ConcurrentModificationException`.
- Corretto un bug di Gremlin che, in determinate condizioni, aggiornando edge o vertici poteva causare un'`InternalFailureException` transitoria.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.2.2.R6, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.8`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.2.2.R6

Se si esegue la versione del motore `1.0.2.2`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.2.2.R6 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.2.2.R5 (12/10/2020)

A partire dal 12 ottobre 2020, viene implementata a livello generale la versione del motore 1.0.2.2.R5. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Miglioramenti in questo rilascio del motore

- Miglioramento delle prestazioni per il passaggio `properties()` di Gremlin.

- Aggiunti dettagli su `BindOp` e `MultiplexerOp` nei rapporti esplicativi e di profilo.
- Per le risposte alle query SPARQL, aggiunto `charset` all'intestazione Content-Type, per consentire ai client HTTP di riconoscere automaticamente il set di caratteri utilizzato.

Difetti corretti in questa versione del motore

- Corretto un bug di SPARQL a causa del quale `CancellationException` non veniva gestita.
- Corretto un bug di SPARQL a causa del quale le query contenenti opzioni nidificate non funzionavano correttamente.
- Corretto un bug di SPARQL in LOAD in cui un'`ConcurrentModificationException` poteva causare il blocco di una query.
- Corretto un bug di SPARQL che impediva di comprimere le risposte alle query con gzip.
- Corretto un bug di Gremlin nel passaggio `groupBy()`.
- Corretto un bug di Gremlin relativo all'uso di un passaggio `aggregate()` all'interno di un passaggio `local()`.
- Corretto un bug di Gremlin relativo all'utilizzo di `bothE()` seguito da un predicato che utilizza valori aggregati.
- Corretto un bug di Gremlin relativo all'utilizzo del passaggio `bothE()` con il passaggio `repeat()`.
- Corretta una potenziale perdita di memoria di Gremlin relativa al passaggio `both()`.
- Corretto un bug in cui mancavano le metriche di richiesta perché un endpoint che terminava con '/' non veniva gestito correttamente.
- Corretto un bug che poteva generare un'`ThrottlingException` anche quando la coda delle richieste non era piena.
- Corretto un bug nel recupero dello stato del caricamento quando un caricamento non riesce per un motivo come `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE`.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.2.2.R5, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.3`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.2.2.R5

Se si esegue la versione del motore `1.0.2.2`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.2.2.R5 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un

aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.2.2.R4 (23/07/2020)

A partire dal 23 luglio 2020, viene implementata a livello generale la versione del motore 1.0.2.2.R4. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Miglioramenti in questo rilascio del motore

- Utilizzo della memoria ottimizzato grazie al rilascio più frequente della memoria inutilizzata al sistema operativo.
- Inoltre, è stato migliorato l'utilizzo della memoria per le query GROUP BY di SPARQL.
- Aumentato il tempo massimo di apertura di una connessione WebSocket autenticata con IAM, da 36 ore a 10 giorni.
- È stata aggiunta la metrica CloudWatch `BufferCacheHitRatio`, che può essere utile per diagnosticare la latenza delle query e ottimizzare i tipi di istanza. Per informazioni, consultare Metriche di Neptune.

Difetti corretti in questa versione del motore

- Corretto un bug nella chiusura di connessioni IAM WebSocket inattive o scadute. Neptune ora invia un frame di chiusura prima di chiudere la connessione.
- Corretto un bug di SPARQL nella valutazione di query contenenti condizioni FILTER EXISTS e/o FILTER NOT EXISTS nidificate.
- Corretto un bug di terminazione delle query SPARQL che causava il blocco dei thread sul server in particolari condizioni critiche.
- Corretto un bug di Gremlin che riguardava Edge pathType nel passaggio `hasLabel`.
- Corretto un bug di Gremlin da gestire `toV` e `fromV` singolarmente per ogni direzione su `bothE`.

- Corretto un bug di Gremlin che comportava la scomparsa di sideEffect.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.2.2.R4, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.3`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.2.2.R4

Se si esegue la versione del motore `1.0.2.2`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.2.2.R4 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.2.2.R3 (22/07/2020)

Il rilascio del motore 1.0.2.2.R3 è stato incorporato nel rilascio del motore 1.0.2.2.R4.

Motore Amazon Neptune versione 1.0.2.2.R2 (02/04/2020)

A partire dal 02 aprile 2020, viene implementata a livello generale la versione del motore 1.0.2.2.R2. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Miglioramenti in questo rilascio del motore

- È ora possibile accodare fino a 64 attività bulk-load, anziché attendere il completamento di una prima di iniziare quella successiva. È inoltre possibile rendere l'esecuzione di una richiesta di caricamento in coda subordinata al completamento di una o più attività di caricamento precedentemente accodate utilizzando il parametro `dependencies` del comando `load`. Per informazioni, consultare Comando dello strumento di caricamento Neptune.
- L'output di ricerca full-text può ora essere ordinato (consulta Parametri per la ricerca full-text).
- È ora disponibile un parametro del cluster database per richiamare i flussi Neptune e la funzionalità è stata spostata fuori dalla modalità di laboratorio. Per informazioni, consultare Abilitazione di Neptune Streams.

Difetti corretti in questa versione del motore

- Corretto un errore stocastico all'avvio del server che ritardava la creazione dell'istanza.
- Corretto un problema di ottimizzazione per il quale le istruzioni `BIND` nella query facevano partire l'ottimizzatore con modelli non selettivi nella pianificazione join-order.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.2.2.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.3`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.2.2.R2

Se si esegue la versione del motore `1.0.2.2`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.2.2.R2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.2.1 (22/11/2019)

Rilascio di patch successive per questa versione

- Rilascio: 1.0.2.1.R6 (22/04/2020)
- Rilascio: 1.0.2.1.R5 (22/04/2020) Questa versione di patch non è stata distribuita.
- Rilascio: 1.0.2.1.R4 (20/12/2019)
- Rilascio: 1.0.2.1.R3 (12/12/2019)
- Rilascio: 1.0.2.1.R2 (25/11/2019)

Nuove caratteristiche in questa versione del motore

- Aggiunte funzionalità di ricerca full-text attraverso l'integrazione con il servizio OpenSearch di Amazon. Per informazioni, consultare Ricerca full-text di Neptune.
- Aggiunta l'opzione che utilizza la modalità di laboratorio per creare un quarto indice (un indice OSGP) per un numero elevato di predicati. Per informazioni, consultare Indice OSGP.

- Aggiunta una modalità dettagli a SPARQL Explain. Consultare Usare SPARQL `explain` e Output con modalità dettagli per i dettagli.
- Aggiunte le informazioni sulla modalità di laboratorio al report sullo stato del motore. Per informazioni dettagliate, vedi Stato dell'istanza.
- Gli snapshot del cluster database ora possono essere copiati nelle regioni AWS. Per informazioni, consultare Copia di uno snapshot.

Miglioramenti in questo rilascio del motore

- Miglioramento delle prestazioni durante la gestione di un numero elevato di predicati.
- Ottimizzazione avanzata delle query. Anche se questo dovrebbe essere completamente trasparente per i clienti, ti invitiamo a testare le tue applicazioni prima di eseguire l'aggiornamento per assicurarti che si comportino come previsto.
- Piccoli miglioramenti alla segnalazione degli errori.
- Aggiunte ottimizzazioni per le fasi Gremlin `.project()` e `.identity()`.
- Aggiunte ottimizzazioni per i casi `.union()` Gremlin non terminali.
- Aggiunto il supporto nativo per gli attraversamenti `.path().by()` di Gremlin.
- Aggiunto il supporto nativo per `.coalesce()` di Gremlin.
- Ulteriore ottimizzazione della scrittura di massa.
- Ora è necessario che le connessioni HTTPS utilizzino almeno la versione TLS 1.2 o superiore, per evitare l'utilizzo di crittografie obsolete/non sicure.

Difetti corretti in questa versione del motore

- Corretto un bug di gestione dell'attraversamento interno `addE()` di Gremlin.
- Corretto un bug di Gremlin causato dalle annotazioni AST che vengono perse durante gli attraversamenti da figlio a padre.
- Corretto un bug che si verificava in Gremlin quando `.otherV()` veniva richiamato dopo `select()`.
- Corretto un bug di Gremlin che causava l'errore di alcune fasi `.hasLabel()` se apparivano dopo una fase `bothE()`.
- Sono state apportate correzioni minori per .sum () e .project () di Gremlin.
- Corretto un bug nell'elaborazione delle query SPARQL senza parentesi graffa di chiusura.

- Corretti alcuni bug minori in SPARQL Explain.
- Corretto un bug nella gestione di richieste simultanee di stato di caricamento.
- Ridotta la memoria utilizzata per eseguire alcuni attraversamenti con le fasi `.project()` di Gremlin.
- Corretti i confronti numerici di valori speciali in SPARQL. Per informazioni, consultare Conformità agli standard.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.2.1, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.1`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.2.1

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Non +possibile effettuare l'aggiornamento automatico a questa versione.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.2.1 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come

lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.2.1.R6 (22/04/2020)

A partire dal 22 aprile 2020, viene implementata a livello generale la versione del motore 1.0.2.1.R6. Tieni presente che occorrono diversi giorni prima che una nuova versione diventi disponibile in ogni regione.

Difetti corretti in questa versione del motore

- Corretto un bug in cui `ConcurrentModificationConflictException` e `TransactionException` non venivano convertite in un'`NeptuneGremlinException`, che causava la restituzione di `InternalFailureException` ai clienti.
- Corretto un bug per il quale Neptune segnalava il suo stato come integro prima che il server fosse completamente pronto.
- Risolto un problema per cui i commit del dizionario e delle transazioni dell'utente erano fuori uso quando due mappature `value->id` venivano inserite contemporaneamente.

- Risolto un bug nella serializzazione dello stato di caricamento.
- Corretto un bug di sessioni di Gremlin.
- Corretto un bug per cui Neptune non riusciva a generare un'eccezione quando il server non riusciva ad avviarsi.
- Corretto un bug a causa del quale Neptune non riusciva a inviare un frame di chiusura WebSocket prima di chiudere il canale.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.2.1.R6, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.1`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.2.1.R6

Se si esegue la versione del motore `1.0.2.1`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.2.1.R6 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
```

```
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come

lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.2.1.R5 (22/04/2020)

Il rilascio 1.0.2.1.R5 del motore non è mai stato implementato.

Motore Amazon Neptune versione 1.0.2.1.R4 (20/12/2019)

Miglioramenti in questo rilascio del motore

- Ora Neptune cerca sempre di posizionare ogni chiamata di ricerca full-text per prima nella pipeline di esecuzione. In questo modo si riduce il volume delle chiamate a OpenSearch, con un significativo miglioramento delle prestazioni. Per informazioni, consultare Esecuzione di query di ricerca full-text.
- Neptune ora genera un'`IllegalArgumentException` se si tenta di accedere a una proprietà inesistente, vertice o edge. In precedenza, Neptune ha sollevato un'`UnsupportedOperationException` in quella situazione.

Ad esempio, se si tenta di aggiungere un edge che fa riferimento a un vertice inesistente, ora si solleva un `IllegalArgumentException`.

Difetti corretti in questa versione del motore

- Corretto un bug Gremlin in cui un attraversamento `union` all'interno di un `project-by` non restituisce risultati o restituisce risultati errati.
- Corretto un bug di Gremlin che causava passaggi nidificati `.project().by()` per restituire risultati non corretti.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.2.1.R4, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.1`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.2.1.R4

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Tuttavia, l'aggiornamento automatico a questa versione non è supportato.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.2.1.R4 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
```

```
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.2.1.R3 (12/12/2019)

Difetti corretti in questa versione del motore

- Corretto un bug in cui l'indice `OSGP` era disabilitato anche se la funzione era correttamente abilitata in Modalità di laboratorio utilizzando il valore `ObjectIndex` nel parametro `neptune_lab_mode`.
- Corretto un bug che interessava le query Gremlin con `.fold()` all'interno di una fase `.project().by()`. A causa di ciò, ad esempio, la seguente query ha restituito risultati incompleti:

  ```
  g.V().project("a").by(valueMap().fold())
  ```

- Corretto un collo di bottiglia delle prestazioni nei caricamenti in blocco di dati RDF.

- Corretto un bug che causava un arresto anomalo delle repliche quando i flussi erano abilitati e la replica veniva riavviata prima della primaria.
- Corretto un bug in cui i certificati SSL ruotati sulle istanze non venivano prelevati senza un riavvio dell'istanza.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.2.1.R3, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.1`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.2.1.R3

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Tuttavia, l'aggiornamento automatico a questa versione non è supportato.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.2.1.R3 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.2.1.R2 (25/11/2019)

Difetti corretti in questa versione del motore

- Risolto un bug che interessava tutte le query `project().by()` non round-robin by-traversal e `path()` non-by-traversal.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.2.1.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.1`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.2.1.R2

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Tuttavia, l'aggiornamento automatico a questa versione non è supportato.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.2.1.R2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.2.0 (08/11/2019)

IMPORTANTE: QUESTA VERSIONE DEL MOTORE È ORA OBSOLETA

A partire dal 19 maggio 2020, non vengono create nuove istanze che utilizzano questa versione del motore.

Questa versione del motore è ora sostituita dalla versione 1.0.2.1, che contiene tutte le correzioni di bug in questa versione, nonché funzionalità aggiuntive come l'integrazione della ricerca full-text, il supporto degli indici OSGP e la copia del cluster snapshot del database nelle regioni AWS.

A partire dal 1° giugno 2020, Neptune aggiorna automaticamente ogni cluster che esegue questa versione del motore alla patch più recente della versione 1.0.2.1 durante la finestra di manutenzione successiva. È possibile eseguire l'aggiornamento manualmente prima di allora, come descritto qui.

In caso di problemi con l'aggiornamento, è possibile contattarci tramite Supporto AWS oppure mediante il forum per sviluppatori AWS.

Rilascio di patch successive per questa versione

- Rilascio: 1.0.2.0.R3 (05/05/2020)
- Rilascio: 1.0.2.0.R2 (21/11/2019)

Nuove caratteristiche in questa versione del motore

Oltre agli aggiornamenti di manutenzione, questa versione aggiunge nuove funzionalità per supportare più versioni del motore contemporaneamente (consultare Gestione del cluster di database Amazon Neptune).

Di conseguenza, la numerazione delle versioni del motore è cambiata (consultare Numerazione delle versioni prima del rilascio del motore 1.3.0.0).

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.2.0, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.1`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.2.0

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Non +possibile effettuare l'aggiornamento automatico a questa versione.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.2.0 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di

inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.2.0.R3 (05/05/2020)

IMPORTANTE: QUESTA VERSIONE DEL MOTORE È ORA OBSOLETA

A partire dal 19 maggio 2020, non vengono create nuove istanze che utilizzano questa versione del motore.

Questa versione del motore è ora sostituita dalla versione 1.0.2.1, che contiene tutte le correzioni di bug in questa versione, nonché funzionalità aggiuntive come l'integrazione della ricerca full-text, il supporto degli indici OSGP e la copia del cluster snapshot del database nelle regioni AWS.

A partire dal 1° giugno 2020, Neptune aggiorna automaticamente ogni cluster che esegue questa versione del motore alla patch più recente della versione 1.0.2.1 durante la finestra di manutenzione successiva. È possibile eseguire l'aggiornamento manualmente prima di allora, come descritto qui.

In caso di problemi con l'aggiornamento, è possibile contattarci tramite Supporto AWS oppure mediante il forum per sviluppatori AWS.

Difetti corretti in questa versione del motore

- Risolto un bug in cui `ConcurrentModificationConflictException` e `TransactionException` sono stati segnalati come generici `InternalFailureException`.
- Corretti i bug nei controlli di integrità che causavano frequenti riavvii del server durante l'avvio.
- Risolto un problema per cui i dati non erano visibili sulle repliche perché i commit erano fuori uso in determinate condizioni.

- Corretto un bug nella serializzazione dello stato di caricamento in cui un carico non riusciva a causa della mancanza di autorizzazioni di accesso ad Amazon S3.
- Corretta una perdita di risorse nelle sessioni di Gremlin.
- Corretto un bug nel controllo dell'integrità che nascondeva lo stato non integro all'avvio dei componenti che gestivano l'autenticazione IAM.
- Corretto un bug a causa del quale Neptune non riusciva a inviare un frame di chiusura WebSocket prima di chiudere il canale.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.2.0.R3, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.1`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.2.0.R3

Se si esegue la versione del motore `1.0.2.0`, il cluster verrà aggiornato automaticamente a questa patch release durante la finestra di manutenzione successiva.

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questa versione.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.2.0.R3 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
```

```
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.2.0.R2 (21/11/2019)

IMPORTANTE: QUESTA VERSIONE DEL MOTORE È ORA OBSOLETA

A partire dal 19 maggio 2020, non vengono create nuove istanze che utilizzano questa versione del motore.

Questa versione del motore è ora sostituita dalla versione 1.0.2.1, che contiene tutte le correzioni di bug in questa versione, nonché funzionalità aggiuntive come l'integrazione della ricerca full-text, il supporto degli indici OSGP e la copia del cluster snapshot del database nelle regioni AWS.

A partire dal 1° giugno 2020, Neptune aggiorna automaticamente ogni cluster che esegue questa versione del motore alla patch più recente della versione 1.0.2.1 durante la finestra di manutenzione successiva. È possibile eseguire l'aggiornamento manualmente prima di allora, come descritto qui.

In caso di problemi con l'aggiornamento, è possibile contattarci tramite Supporto AWS oppure mediante il forum per sviluppatori AWS.

Difetti corretti in questa versione del motore

- Migliorata la strategia di memorizzazione nella cache per le pagine sporche sul server in modo che `FreeableMemory` venga ripristinato più velocemente quando il server entra in uno stato di memoria insufficiente.
- Corretto un bug che poteva causare una race condition e un arresto quando il server elabora molte richieste contemporanee di stato di caricamento e/o richieste di avvio del caricamento.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.2.0.R2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.1`
- Versione di SPARQL: `1.1`

Percorsi di aggiornamento al rilascio del motore 1.0.2.0.R2

È possibile aggiornare manualmente qualsiasi rilascio del motore Neptune precedente a questo rilascio.

Tuttavia, l'aggiornamento automatico a questa versione non è supportato.

Aggiornamento a questo rilascio

Amazon Neptune 1.0.2.0.R2 è ora disponibile a livello generale.

Se un cluster database utilizza una versione del motore dalla quale esiste un percorso di aggiornamento a questo rilascio, ora è idoneo all'aggiornamento. È possibile aggiornare qualsiasi cluster idoneo utilizzando le operazioni del cluster database sulla console o utilizzando SDK. Il seguente comando CLI aggiornerà immediatamente un cluster idoneo:

Per Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
```

```
    --apply-immediately
```

Per Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster di database. Un aggiornamento richiede il riavvio del database su queste istanze, quindi si verificheranno tempi di inattività che vanno da 20-30 secondi a diversi minuti, dopodiché si potrà riprendere a usare il cluster database.

Eseguire sempre un test prima dell'aggiornamento

Quando viene rilasciata una nuova versione principale o secondaria del motore Neptune, testa sempre le applicazioni Neptune su di essa prima di procedere all'aggiornamento. Anche un aggiornamento secondario potrebbe introdurre nuove funzionalità o comportamenti che possono influire sul codice.

Inizia confrontando le pagine delle note di rilascio della versione corrente con quelle della versione di destinazione per valutare se verranno modificate le versioni del linguaggio di query o verranno introdotte altre modifiche che causano interruzioni.

Il modo migliore per testare una nuova versione prima di aggiornare il cluster database di produzione è clonare il cluster di produzione affinché il clone esegua la nuova versione del motore. È quindi possibile eseguire query sul clone senza influire sul cluster database di produzione.

Creare sempre uno snapshot manuale prima dell'aggiornamento

Prima di procedere a un aggiornamento, è consigliabile creare sempre uno snapshot manuale del cluster database. Uno snapshot automatico offre solo una protezione a breve termine, mentre uno snapshot manuale rimane disponibile fino a quando non lo elimini esplicitamente.

In alcuni casi Neptune crea automaticamente uno snapshot manuale come parte del processo di aggiornamento, ma non è consigliabile farvi affidamento ed è comunque opportuno creare sempre il proprio snapshot manuale.

Quando hai la certezza che non sarà necessario ripristinare lo stato precedente all'aggiornamento del cluster di database, puoi eliminare in modo esplicito lo snapshot manuale che hai creato, così come

lo snapshot manuale eventualmente creato da Neptune. Se Neptune crea uno snapshot manuale, questo avrà un nome che inizia con `preupgrade`, seguito dal nome del cluster database, dalla versione del motore di origine, dalla versione del motore di destinazione e dalla data.

> **Note**
>
> Se stai tentando di eseguire l'aggiornamento mentre è in corso un'azione in sospeso, potrebbe verificarsi un errore come il seguente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Se riscontri questo errore, attendi il completamento dell'azione in sospeso o attiva immediatamente una finestra di manutenzione per completare l'aggiornamento precedente.

Per ulteriori informazioni sull'aggiornamento della versione del motore, consulta Gestione del cluster di database Amazon Neptune. In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Motore Amazon Neptune versione 1.0.1.2 (10/06/2020)

IMPORTANTE: QUESTA VERSIONE DEL MOTORE È ORA OBSOLETA

A partire dal 27 aprile 2021, non vengono create nuove istanze che utilizzano questa versione del motore.

Miglioramenti in questo rilascio del motore

- Neptune ora genera un'`IllegalArgumentException` se si tenta di accedere a una proprietà inesistente, vertice o edge. In precedenza, Neptune ha sollevato un'`UnsupportedOperationException` in quella situazione.

  Ad esempio, se si tenta di aggiungere un edge che fa riferimento a un vertice inesistente, ora si solleva un'`IllegalArgumentException`.

Difetti corretti in questa versione del motore

- Corretto un bug per cui i commit del dizionario e delle transazioni dell'utente erano fuori uso quando due mappature `value->id` venivano inserite contemporaneamente.
- Corretto un bug nella serializzazione dello stato di caricamento.
- Corretto un errore stocastico all'avvio del server che ritardava la creazione dell'istanza.
- Corretta una perdita di cursore.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.1.2, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.4.1`
- Versione di SPARQL: `1.1`

Motore Amazon Neptune versione 1.0.1.1 (26/06/2020)

IMPORTANTE: QUESTA VERSIONE DEL MOTORE È ORA OBSOLETA

A partire dal 27 aprile 2021, non vengono create nuove istanze che utilizzano questa versione del motore.

Difetti corretti in questa versione del motore

- Risolto un problema per cui i commit erano fuori uso quando inseriti contemporaneamente.
- Corretto un bug nella serializzazione dello stato di caricamento.
- Corretto un errore stocastico all'avvio del server che ritardava la creazione dell'istanza.
- Corretta una perdita di memoria.

Versioni di linguaggio di query supportate in questo rilascio

Prima di aggiornare un cluster database alla versione 1.0.1.1, assicurati che il tuo progetto sia compatibile con queste versioni di linguaggio di query:

- Versione di Gremlin: `3.3.2`
- Versione di SPARQL: `1.1`

Motore Amazon Neptune versione 1.0.1.0 (02/07/2019)

IMPORTANTE: QUESTA VERSIONE DEL MOTORE È ORA OBSOLETA

A partire dal 27 aprile 2021, non vengono create nuove istanze che utilizzano questa versione del motore.

Aggiornamenti del motore Amazon Neptune 31/10/2019

Versione: 1.0.1.0.200502.0

IMPORTANTE: QUESTA VERSIONE DEL MOTORE È ORA OBSOLETA

A partire dal 27 aprile 2021, non vengono create nuove istanze che utilizzano questa versione del motore.

Difetti corretti in questa versione del motore

- Risolto un bug Gremlin nella serializzazione della risposta del passo `tree()` quando i client si connettono a Neptune utilizzando `traversal().withRemote(...)` (in altre parole, utilizzando bytecode GLV).

  Questa versione risolve un problema in cui i client che si sono connessi a Neptune utilizzando `traversal().withRemote(...)` hanno ricevuto una risposta non valida alle query Gremlin che contenevano un passaggio `tree()`.
- Risolto un bug SPARQL nelle query `DELETE WHERE LIMIT`, in cui il processo di terminazione della query si bloccherebbe a causa di una race condition, causando il timeout della query.

Aggiornamenti del motore Amazon Neptune 15/10/2019

Versione: 1.0.1.0.200463.0

IMPORTANTE: QUESTA VERSIONE DEL MOTORE È ORA OBSOLETA

A partire dal 27 aprile 2021, non vengono create nuove istanze che utilizzano questa versione del motore.

Nuove caratteristiche in questa versione del motore

- È stata aggiunta una caratteristica Explain/Profile di Gremlin (consulta Analisi dell'esecuzione di query Neptune tramite la funzionalità Gremlin `explain`).
- È stato aggiunto Supporto delle sessioni basate su script Gremlin per abilitare l'esecuzione di più attraversamenti Gremlin in una singola transazione.
- Aggiunto il supporto per l'estensione della query federata SPARQL in Neptune (consulta SPARQL 1.1 Federated Query e Query federate SPARQL in Neptune che utilizzano l'estensione `SERVICE`).
- È stata aggiunta una caratteristica che consente di inserire un proprio `queryId` in una query Gremlin o SPARQL, tramite un parametro URL HTTP o tramite un hint di query `queryId` SPARQL (consulta Inserimento di un ID personalizzato in una query Neptune Gremlin o SPARQL).
- È stata aggiunta una funzionalità Modalità di laboratorio in Neptune che consente di provare nuove funzionalità che non sono ancora pronte per essere utilizzate in produzione.
- È stata aggiunta una nuova caratteristica Neptune Streams che registra in modo affidabile ogni modifica apportata al database in un flusso che persiste per una settimana. Questa caratteristica è disponibile solo in modalità di laboratorio.
- È stata aggiornata la semantica formale per transazioni simultanee (consulta Semantica delle transazioni in Neptune). Questa caratteristica fornisce garanzie standard di settore sulla concorrenza.

  Per impostazione predefinita, questa semantica delle transazioni è abilitata. In alcuni scenari, questa caratteristica potrebbe modificare il comportamento di caricamento corrente e ridurre le prestazioni di caricamento. Puoi utilizzare il parametro `neptune_lab_mode` del cluster di database per ripristinare la semantica precedente includendo `ReadWriteConflictDetection=disabled` nel valore del parametro.

Miglioramenti in questo rilascio del motore

- È stata migliorata l'API Stato dell'istanza mediante la segnalazione della versione di TinkerPop e la versione di SPARQL utilizzata dal motore.
- Sono state migliorate le prestazioni dell'operatore del grafo secondario Gremlin.
- Sono state migliorate le prestazioni di serializzazione risposta Gremlin.
- Sono state migliorate le prestazioni nella fase Union Gremlin.
- È stata migliorata la latenza di semplici query SPARQL.

Difetti corretti in questa versione del motore

- È stato corretto un bug Gremlin per cui il timeout veniva restituito erroneamente come errore interno.
- È stato corretto un bug SPARQL in cui ORDER BY su un set parziale di variabili generava un errore interno del server.

Aggiornamenti del motore Amazon Neptune 19/09/2019

IMPORTANTE: QUESTA VERSIONE DEL MOTORE È ORA OBSOLETA

A partire dal 27 aprile 2021, non vengono create nuove istanze che utilizzano questa versione del motore.

Versione: 1.0.1.0.200457.0

Amazon Neptune 1.0.1.0.200457.0 è disponibile a livello generale. Al termine dell'aggiornamento del motore per tale regione, tutti i nuovi cluster database di Neptune, inclusi quelli ripristinati dagli snapshot, vengono creati nella versione 1.0.1.0.200457.0.

Puoi aggiornare immediatamente a questa versione i cluster esistenti mediante le operazioni sui cluster database nella console o tramite l'SDK. È possibile utilizzare il seguente comando della CLI per aggiornare un cluster DB:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster database. Un aggiornamento richiede un riavvio del database per tutte le istanze di un cluster database, pertanto ci sarà un tempo di inattività di 20-30 secondi, al termine del quale sarà possibile ricominciare a utilizzare il cluster o i cluster database. Per visualizzare o modificare le impostazioni della finestra di manutenzione, accedi alla console Neptune.

In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Difetti corretti in questa versione del motore

- È stato risolto un problema di correttezza di Gremlin introdotto nella versione precedente del motore (1.0.1.200369.0) rimuovendo il miglioramento delle prestazioni della gestione congiunta dei predicati che lo causava.
- È stato corretto un bug di SPARQL che causava la generazione di un query con `DISTINCT` e un singolo modello incluso in `OPTIONAL` per generare un `InternalServerError`.

Aggiornamenti del motore Amazon Neptune 13/08/2019

IMPORTANTE: QUESTA VERSIONE DEL MOTORE È ORA OBSOLETA

A partire dal 27 aprile 2021, non vengono create nuove istanze che utilizzano questa versione del motore.

Nuove caratteristiche in questa versione del motore

- È stata aggiunta un'opzione `OVERSUBSCRIBE` al parametro `parallelism` di Comando dello strumento di caricamento Neptune, che fa sì che lo strumento di caricamento in blocco Neptune utilizzi tutti i thread e le risorse disponibili.

Miglioramenti in questo rilascio del motore

- Sono state migliorate le prestazioni dei filtri SPARQL contenenti semplici espressioni OR logiche.
- Sono state migliorate le prestazioni Gremlin nella gestione dei predicati congiuntivi.

Difetti corretti in questa versione del motore

- È stato corretto un bug di SPARQL che impediva la sottrazione di un `xsd:duration` da un `xsd:date`.
- È stato corretto un bug SPARQL che causava risultati incompleti dall'inlining statico in presenza di un `UNION`.
- È stato corretto un bug SPARQL nell'annullamento della query.
- È stato corretto un bug Gremlin che causava l'overflow durante la promozione del tipo.
- È stato corretto un bug Gremlin nella gestione degli elementi dei vertici nelle fasi `addE().from().to()`.

- È stato corretto un bug Gremlin (rilasciato il 26 luglio 2019 nella versione del motore 1.0.1.200366.0) che comportava la gestione dei doppi NaN e dei float in inserimenti a cardinalità singola.
- È stato corretto un bug nella generazione di piani di query che comportavano ricerche basate su proprietà.

Aggiornamenti del motore Amazon Neptune 26/07/2019

Versione: 1.0.1.0.200366.0

IMPORTANTE: QUESTA VERSIONE DEL MOTORE È ORA OBSOLETA

A partire dal 27 aprile 2021, non vengono create nuove istanze che utilizzano questa versione del motore.

Nuove caratteristiche in questa versione del motore

- Aggiornato a TinkerPop 3.4.1 (consulta Informazioni sull'aggiornamento TinkerPop e Log modifiche TinkerPop 3.4.1).

  Per i clienti di Neptune, queste modifiche forniscono nuove funzionalità e miglioramenti, ad esempio:

  - `GraphBinary` è ora disponibile come formato di serializzazione.
  - Un bug keep-alive che causava perdite di memoria nel driver Java TinkerPop è stato corretto, pertanto non è più necessaria una soluzione alternativa.

  Tuttavia, in alcuni casi, possono influire sul codice Gremlin esistente in Neptune. Ad esempio:

  - `valueMap()` ora restituisce un `Map<Object,Object>` invece di un `Map<String,Object>`.
  - Il comportamento incoerente della fase `within()` è stato corretto in modo che funzionasse in modo coerente con altre fasi. In precedenza, i tipi dovevano corrispondere affinché i confronti funzionassero. Ora, i numeri di tipi diversi possono essere confrontati con precisione. Ad esempio, a differenza del passato, `33` ora è uguale a `33L`.
  - Un bug in `ReducingBarrierStep` è stato corretto, quindi ora non restituisce alcun valore se non sono disponibili elementi per l'output.
  - L'ordine degli ambiti `select()` è stato modificato (l'ordine è ora `maps`, `side-effects`, `paths`). Questo modifica i risultati delle query rare che combinano `side-effects` e `select` con lo stesso nome di chiave per `side-effects` come per `select`.

  - `bulkSet()` ora fa parte del protocollo GraphSON. Le query che terminano con `toBulkSet()` non funzioneranno con i client meno recenti.
  - Una parametrizzazione della fase `Submit()` è stata rimossa dal client 3.4.

  Molte altre modifiche introdotte in TinkerPop 3.4 non influiscono sul comportamento corrente di Neptune. Ad esempio, `io()` Gremlin è stato aggiunto come fase per `Traversal` ed è ora obsoleto in `Graph`, ma non è mai stato abilitato in Neptune.
- Aggiunto il supporto per proprietà del vertice a cardinalità singola allo strumento di caricamento in blocco per Gremlin, per il caricamento dei dati di un grafico di proprietà.
- Aggiunta un'opzione per sovrascrivere i valori esistenti per una proprietà a cardinalità singola nello strumento di caricamento in blocco.
- Aggiunta la possibilità di recuperare lo stato di una query Gremlin e di annullare una query Gremlin.
- Aggiunto un hint di query per timeout delle query SPARQL.
- Aggiunta la possibilità di visualizzare il ruolo dell'istanza nell'API di stato (consulta Stato dell'istanza).
- Aggiunto il supporto per la clonazione del database (consulta Clonazione del database in Neptune).

Miglioramenti in questo rilascio del motore

- Migliorata la spiegazione query SPARQL per mostrare variabili del grafico da clausole FROM.
- Prestazioni migliorate per SPARQL in filtri, filtri di uguaglianza, clausole VALUES e conteggi intervallo.
- Sono state migliorate le prestazioni per l'ordinamento delle fasi Gremlin.
- Migliorate prestazioni per attraversamenti `.repeat.dedup` Gremlin.
- Migliorate le prestazioni di attraversamenti `valueMap()` e `path().by()` Gremlin.

Difetti corretti in questa versione del motore

- Corretti più problemi con i percorsi di proprietà SPARQL, incluse operazioni con grafici denominati.
- Corretto un bug con le query SPARQL CONSTRUCT che causa problemi di memoria.
- Corretto un bug con il parser RDF Turtle e i nomi locali.
- Corretto un bug per correggere messaggi di errore visualizzati agli utenti.

- Corretto un bug con attraversamenti `repeat()...drop()` Gremlin.
- Corretto un bug con la fase `drop()` Gremlin.
- Corretto un bug con i filtri etichetta Gremlin.
- Corretto un bug con i timeout delle query Gremlin.

Aggiornamenti del motore Amazon Neptune 02/07/2019

IMPORTANTE: QUESTA VERSIONE DEL MOTORE È ORA OBSOLETA

A partire dal 27 aprile 2021, non vengono create nuove istanze che utilizzano questa versione del motore.

Difetti corretti in questa versione del motore

- Corretto un bug che causava la mancata ottimizzazione di determinati modelli con un nome di proprietà e un limite di valore.

Rilasci precedenti del motore Neptune

Argomenti

Aggiornamenti del motore Amazon Neptune 12/06/2019

Versione: 1.0.1.0.200310.0

Amazon Neptune 1.0.1.0.200310.0 è disponibile a livello generale. Al termine dell'aggiornamento del motore per tale regione, tutti i nuovi cluster database di Neptune, inclusi quelli ripristinati dagli snapshot, vengono creati nella versione 1.0.1.0.200310.0.

Puoi aggiornare immediatamente a questa versione i cluster esistenti mediante le operazioni sui cluster database nella console o tramite l'SDK. È possibile utilizzare il seguente comando dell'interfaccia CLI per aggiornare immediatamente un cluster database a questo rilascio:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

I cluster database di Neptune vengono aggiornati automaticamente al rilascio del motore 1.0.1.0.200310.0 durante le finestre di manutenzione del sistema. I tempi per l'applicazione degli aggiornamenti dipendono dalla regione e dall'impostazione della finestra di manutenzione del cluster database, ma anche dal tipo di aggiornamento.

> **Note**
>
> La finestra di manutenzione dell'istanza non si applica agli aggiornamenti del motore.

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster database. Un aggiornamento richiede un riavvio del database per tutte le istanze di un cluster database, pertanto ci sarà un tempo di inattività di 20-30 secondi, al termine del quale sarà possibile ricominciare a utilizzare il cluster o i cluster database. Per visualizzare o modificare le impostazioni della finestra di manutenzione, accedi alla console Neptune.

In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Miglioramenti

- Corregge un bug in cui inserimenti e rimozioni simultanee di una edge possono generare più edge con lo stesso ID.
- Altre correzioni e miglioramenti minori.

Aggiornamenti del motore Amazon Neptune 01/05/2019

Versione: 1.0.1.0.200296.0

Amazon Neptune 1.0.1.0.200296.0 è disponibile a livello generale. Al termine dell'aggiornamento del motore per tale regione, tutti i nuovi cluster database di Neptune, inclusi quelli ripristinati dagli snapshot, vengono creati nella versione 1.0.1.0.200296.0.

Puoi aggiornare immediatamente a questa versione i cluster esistenti mediante le operazioni sui cluster database nella console o tramite l'SDK. È possibile utilizzare il seguente comando dell'interfaccia CLI per aggiornare immediatamente un cluster database a questo rilascio:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

I cluster database di Neptune vengono aggiornati automaticamente al rilascio del motore 1.0.1.0.200296.0 durante le finestre di manutenzione del sistema. I tempi per l'applicazione degli aggiornamenti dipendono dalla regione e dall'impostazione della finestra di manutenzione del cluster database, ma anche dal tipo di aggiornamento.

> **Note**
>
> La finestra di manutenzione dell'istanza non si applica agli aggiornamenti del motore.

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster database. Un aggiornamento richiede un riavvio del database per tutte le istanze di un cluster database, pertanto ci sarà un tempo di inattività di 20-30 secondi, al termine del quale sarà possibile ricominciare a utilizzare il cluster o i cluster database. Per visualizzare o modificare le impostazioni della finestra di manutenzione, accedi alla console Neptune.

In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Miglioramenti

- È stata aggiunta la nuova funzionalità `explain` alle query SPARQL di Neptune per visualizzare il piano di query e seguire le fasi per ottimizzarlo se necessario. Per informazioni, consultare SPARQL `explain`.
- Migliorate le prestazioni e la creazione di report di SPARQL in diversi modi.
- Migliorate le prestazioni e il comportamento di Gremlin in diversi modi.
- Migliorato il timeout di query `drop( )` con tempi di esecuzione lunghi.
- Migliorate le prestazioni delle query `otherV( )`.
- Sono stati aggiunti due campi alle informazioni restituite quando si interroga lo stato di un'istanza o un cluster database, ovvero il numero di versione del motore e l'ora di avvio del cluster o dell'istanza. Per informazioni, consultare Stato dell'istanza.
- L'API `Get-Status` del loader di Neptune ora restituisce un campo `startTime` che registra il momento di avvio di un processo.
- Il comando dello strumento di caricamento ora accetta un parametro `parallelism` facoltativo che ti permette di limitare il numero di thread utilizzati dallo strumento di caricamento.

Aggiornamenti del motore Amazon Neptune 21/01/2019

Versione: 1.0.1.0.200267.0

Amazon Neptune 1.0.1.0.200267.0 è disponibile a livello generale. Al termine dell'aggiornamento del motore per tale regione, tutti i nuovi cluster database di Neptune, inclusi quelli ripristinati dagli snapshot, vengono creati nella versione 1.0.1.0.200267.0.

Puoi aggiornare immediatamente a questa versione i cluster esistenti mediante le operazioni sui cluster database nella console o tramite l'SDK. È possibile utilizzare il seguente comando dell'interfaccia CLI per aggiornare immediatamente un cluster database a questo rilascio:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

I cluster database di Neptune vengono aggiornati automaticamente al rilascio del motore 1.0.1.0.200267.0 durante le finestre di manutenzione del sistema. I tempi per l'applicazione degli

aggiornamenti dipendono dalla regione e dall'impostazione della finestra di manutenzione del cluster database, ma anche dal tipo di aggiornamento.

> **Note**
>
> La finestra di manutenzione dell'istanza non si applica agli aggiornamenti del motore.

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster database. Un aggiornamento richiede un riavvio del database per tutte le istanze di un cluster database, pertanto ci sarà un tempo di inattività di 20-30 secondi, al termine del quale sarà possibile ricominciare a utilizzare il cluster o i cluster database. Per visualizzare o modificare le impostazioni della finestra di manutenzione, accedi alla console Neptune.

In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Miglioramenti

- Neptune attende più a lungo (entro il timeout di query specificato) la risoluzione di eventuali conflitti. In questo modo viene ridotto il numero di eccezioni di modifica simultanee che il client deve gestire (consulta Errori di query).
- Risolto un problema per cui l'applicazione della cardinalità di Gremlin talvolta causava il riavvio del motore.
- Miglioramento delle prestazioni di Gremlin per le query ripetute di `emit.times`.
- Risoluzione di un problema in Gremlin per cui `repeat.until` lasciava passare soluzioni `.emit` che invece avrebbero dovuto essere filtrate.
- Miglioramento della gestione degli errori in Gremlin.

Aggiornamenti del motore Amazon Neptune 19/11/2018

Versione: 1.0.1.0.200264.0

Amazon Neptune 1.0.1.0.200264.0 è disponibile a livello generale. Al termine dell'aggiornamento del motore per tale regione, tutti i nuovi cluster database di Neptune, inclusi quelli ripristinati dagli snapshot, vengono creati nella versione 1.0.1.0.200264.0.

Puoi aggiornare immediatamente a questa versione i cluster esistenti mediante le operazioni sui cluster database nella console o tramite l'SDK. È possibile utilizzare il seguente comando dell'interfaccia CLI per aggiornare immediatamente un cluster database a questo rilascio:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

I cluster database di Neptune vengono aggiornati automaticamente al rilascio del motore 1.0.1.0.200264.0 durante le finestre di manutenzione del sistema. I tempi per l'applicazione degli aggiornamenti dipendono dalla regione e dall'impostazione della finestra di manutenzione del cluster database, ma anche dal tipo di aggiornamento.

> **Note**
>
> La finestra di manutenzione dell'istanza non si applica agli aggiornamenti del motore.

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster database. Un aggiornamento richiede un riavvio del database per tutte le istanze di un cluster database, pertanto ci sarà un tempo di inattività di 20-30 secondi, al termine del quale sarà possibile ricominciare a utilizzare il cluster o i cluster database. Per visualizzare o modificare le impostazioni della finestra di manutenzione, accedi alla console Neptune.

In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Miglioramenti

- Aggiunto il supporto per the section called “Hint di query”.
- Messaggi di errore migliorati per l'autenticazione IAM. Per ulteriori informazioni, consulta the section called “Errori IAM”.
- Prestazioni delle query SPARQL migliorate con un numero elevato di predicati.
- Miglioramento delle prestazioni del percorso delle proprietà SPARQL.
- Miglioramento delle prestazioni Gremlin per le mutazioni condizionali, ad esempio il modello `fold().coalesce(unfold(), …)`, quando usate con le fasi `addV()`, `property()` e `addE()`.
- Miglioramento delle prestazioni Gremlin per le modulazioni `by()` e `sack()`.

- Miglioramento delle prestazioni Gremlin per le fasi `group()` e `groupCount()`.
- Miglioramento delle prestazioni Gremlin per le fasi `store()`, `sideEffect()` e `cap().unfold()`.
- Supporto migliorato per i vincoli delle proprietà di cardinalità singola Gremlin.
  - Migliore applicazione della cardinalità singola per le proprietà edge e le proprietà vertex contrassegnate come proprietà di cardinalità singola.
  - È stato introdotto un errore se vengono specificati valori di proprietà aggiuntivi per una proprietà edge esistente durante i processi di caricamento Neptune.

Aggiornamenti del motore Amazon Neptune 08/11/2018

Versione: 1.0.1.0.200258.0

Amazon Neptune 1.0.1.0.200258.0 è disponibile a livello generale. Al termine dell'aggiornamento del motore per tale regione, tutti i nuovi cluster database di Neptune, inclusi quelli ripristinati dagli snapshot, vengono creati nella versione 1.0.1.0.200258.0.

Puoi aggiornare immediatamente a questa versione i cluster esistenti mediante le operazioni sui cluster database nella console o tramite l'SDK. È possibile utilizzare il seguente comando dell'interfaccia CLI per aggiornare immediatamente un cluster database a questo rilascio:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

I cluster database di Neptune vengono aggiornati automaticamente al rilascio del motore 1.0.1.0.200258.0 durante le finestre di manutenzione del sistema. I tempi per l'applicazione degli aggiornamenti dipendono dalla regione e dall'impostazione della finestra di manutenzione del cluster database, ma anche dal tipo di aggiornamento.

> **Note**
>
> La finestra di manutenzione dell'istanza non si applica agli aggiornamenti del motore.

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster database. Un aggiornamento richiede un riavvio del database per tutte le istanze di un cluster database,

pertanto ci sarà un tempo di inattività di 20-30 secondi, al termine del quale sarà possibile ricominciare a utilizzare il cluster o i cluster database. Per visualizzare o modificare le impostazioni della finestra di manutenzione, accedi alla console Neptune.

In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Miglioramenti

- Aggiunto il supporto per Hint di query SPARQL.
- Prestazioni migliorate per le query SPARQL FILTER (NOT) Exists.
- Prestazioni migliorate per le query SPARQL DESCRIBE.
- Prestazioni migliorate per la ripetizione fino al modello in Gremlin.
- Prestazioni migliorate per l'aggiunta di edge in Gremlin.
- Risolto un problema a causa del quale le query SPARQL Update DELETE in alcuni casi non hanno esito positivo.
- Risolto un problema per la gestione dei timeout con il server Gremlin WebSocket.

Aggiornamenti del motore Amazon Neptune 29/10/2018

Versione: 1.0.1.0.200255.0

Amazon Neptune 1.0.1.0.200255.0 è disponibile a livello generale. Al termine dell'aggiornamento del motore per tale regione, tutti i nuovi cluster database di Neptune, inclusi quelli ripristinati dagli snapshot, vengono creati nella versione 1.0.1.0.200255.0.

Puoi aggiornare immediatamente a questa versione i cluster esistenti mediante le operazioni sui cluster database nella console o tramite l'SDK. È possibile utilizzare il seguente comando dell'interfaccia CLI per aggiornare immediatamente un cluster database a questo rilascio:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

I cluster database di Neptune vengono aggiornati automaticamente al rilascio del motore 1.0.1.0.200255.0 durante le finestre di manutenzione del sistema. I tempi per l'applicazione degli

aggiornamenti dipendono dalla regione e dall'impostazione della finestra di manutenzione del cluster database, ma anche dal tipo di aggiornamento.

> **Note**
>
> La finestra di manutenzione dell'istanza non si applica agli aggiornamenti del motore.

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster database. Un aggiornamento richiede un riavvio del database per tutte le istanze di un cluster database, pertanto ci sarà un tempo di inattività di 20-30 secondi, al termine del quale sarà possibile ricominciare a utilizzare il cluster o i cluster database. Per visualizzare o modificare le impostazioni della finestra di manutenzione, accedi alla console Neptune.

In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Miglioramenti

- Sono state aggiunte le informazioni di autenticazione IAM ai log di audit.
- Aggiunto il supporto per le credenziali temporanee utilizzando i ruoli IAM e i profili di istanza.
- È stata aggiunta la chiusura della connessione WebSocket per l'autenticazione IAM quando viene revocata l'autorizzazione o se l'utente o il ruolo IAM viene eliminato.
- È stato limitato il numero massimo di connessioni WebSocket a 60.000 per istanza.
- Prestazioni di carico in blocco ottimizzate per i tipi di istanza più piccoli.
- Prestazioni migliorate per le query che includono gli operatori `and()`, `or()`, `not()`, `drop()` in Gremlin.
- Il parser NTriples ora rifiuta gli URI non validi, ad esempio gli URI contenenti spazi.

Aggiornamenti del motore Amazon Neptune 06/09/2018

Versione: 1.0.1.0.200237.0

Amazon Neptune 1.0.1.0.200237.0 è disponibile a livello generale. Al termine dell'aggiornamento del motore per tale regione, tutti i nuovi cluster database di Neptune, inclusi quelli ripristinati dagli snapshot, vengono creati nella versione 1.0.1.0.200237.0.

Puoi aggiornare immediatamente a questa versione i cluster esistenti mediante le operazioni sui cluster database nella console o tramite l'SDK. È possibile utilizzare il seguente comando dell'interfaccia CLI per aggiornare immediatamente un cluster database a questo rilascio:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

I cluster database di Neptune vengono aggiornati automaticamente al rilascio del motore 1.0.1.0.200237.0 durante le finestre di manutenzione del sistema. I tempi per l'applicazione degli aggiornamenti dipendono dalla regione e dall'impostazione della finestra di manutenzione del cluster database, ma anche dal tipo di aggiornamento.

> **Note**
>
> La finestra di manutenzione dell'istanza non si applica agli aggiornamenti del motore.

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster database. Un aggiornamento richiede un riavvio del database per tutte le istanze di un cluster database, pertanto ci sarà un tempo di inattività di 20-30 secondi, al termine del quale sarà possibile ricominciare a utilizzare il cluster o i cluster database. Per visualizzare o modificare le impostazioni della finestra di manutenzione, accedi alla console Neptune.

In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Miglioramenti

- È stato risolto un problema a causa del quale alcune query `SPARQL COUNT(DISTINCT)` non andavano a buon fine.
- È stato risolto un problema a causa del quale le query `COUNT`, `SUM`, `MIN` con una clausola `DISTINCT` esaurivano la memoria.
- È stato risolto un problema a causa del quale i dati di tipo `BLOB` determinavano un errore del processo dello strumento di caricamento Neptune.
- È stato risolto un problema a causa del quale inserimenti duplicati generavano errori di transizione.
- È stato risolto un problema a causa del quale query `DROP ALL` non potevano essere annullate.

- È stato risolto un problema a causa del quale i client Gremlin potevano interrompersi in maniera intermittente.
- Tutti i codici di errore per payload più grandi di 150 M sono stati aggiornati a `HTTP 400`.
- Sono state migliorate le prestazioni e la precisione di query `COUNT()` single-triple-pattern.
- Sono state migliorate le prestazioni di query `SPARQL UNION` con clausole `BIND`.

Aggiornamenti del motore Amazon Neptune 24/07/2018

Versione: 1.0.1.0.200236.0

Amazon Neptune 1.0.1.0.200236.0 è disponibile a livello generale. Al termine dell'aggiornamento del motore per tale regione, tutti i nuovi cluster database di Neptune, inclusi quelli ripristinati dagli snapshot, vengono creati nella versione 1.0.1.0.200236.0.

Puoi aggiornare immediatamente a questa versione i cluster esistenti mediante le operazioni sui cluster database nella console o tramite l'SDK. È possibile utilizzare il seguente comando dell'interfaccia CLI per aggiornare immediatamente un cluster database a questo rilascio:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

I cluster database di Neptune vengono aggiornati automaticamente al rilascio del motore 1.0.1.0.200236.0 durante le finestre di manutenzione del sistema. I tempi per l'applicazione degli aggiornamenti dipendono dalla regione e dall'impostazione della finestra di manutenzione del cluster database, ma anche dal tipo di aggiornamento.

> **Note**
>
> La finestra di manutenzione dell'istanza non si applica agli aggiornamenti del motore.

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster database. Un aggiornamento richiede un riavvio del database per tutte le istanze di un cluster database, pertanto ci sarà un tempo di inattività di 20-30 secondi, al termine del quale sarà possibile ricominciare a utilizzare il cluster o i cluster database. Per visualizzare o modificare le impostazioni della finestra di manutenzione, accedi alla console Neptune.

In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Miglioramenti

- Serializzazione SPARQL per il tipo di dati `xsd:string` è stata aggiornata. xsd:string non è più incluso nella serializzazione JSON, che è ora coerente con altri formati di output.
- La gestione di `xsd:double`/`xsd:float` infinito è stata corretta. I valori `-INF`, `NaN` e `INF` sono ora riconosciuti e gestiti correttamente in tutti i formati dello strumento di caricamento dei dati SPARQL, SPARQL 1.1 UPDATE e SPARQL 1.1 Query.
- Risolto un problema per cui una query Gremlin con valori di stringa vuoti riporta inaspettatamente esito negativo.
- Risolto un problema per cui Gremlin `aggregate()` e `cap()` su un grafico vuoto riportano un errore imprevisto.
- Risolto un problema per cui le risposte di errore errate vengono restituite per Gremlin in caso di specifica di cardinalità non valida, ad esempio `.property(set,id,'10')` e `.property(single,id,'10')`.
- Risolto un problema per cui una sintassi Gremlin non valida viene restituita come InternalFailureException.
- Corretta l'ortografia in `TimeLimitExceeededException` su `TimeLimitExceededException`, nei messaggi di errore.
- Gli endpoint GREMLIN e SPARQL modificati rispondono in modo coerente quando non viene fornito uno script.
- Messaggi di errore chiariti per un numero eccessivo di richieste simultanee.

Aggiornamenti del motore Amazon Neptune 22/06/2018

Versione: 1.0.1.0.200233.0

Amazon Neptune 1.0.1.0.200233.0 è disponibile a livello generale. Al termine dell'aggiornamento del motore per tale regione, tutti i nuovi cluster database di Neptune, inclusi quelli ripristinati dagli snapshot, vengono creati nella versione 1.0.1.0.200233.0.

Puoi aggiornare immediatamente a questa versione i cluster esistenti mediante le operazioni sui cluster database nella console o tramite l'SDK. È possibile utilizzare il seguente comando dell'interfaccia CLI per aggiornare immediatamente un cluster database a questo rilascio:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

I cluster database di Neptune vengono aggiornati automaticamente al rilascio del motore 1.0.1.0.200233.0 durante le finestre di manutenzione del sistema. I tempi per l'applicazione degli aggiornamenti dipendono dalla regione e dall'impostazione della finestra di manutenzione del cluster database, ma anche dal tipo di aggiornamento.

> **Note**
>
> La finestra di manutenzione dell'istanza non si applica agli aggiornamenti del motore.

Gli aggiornamenti vengono applicati contemporaneamente a tutte le istanze in un cluster database. Un aggiornamento richiede un riavvio del database per tutte le istanze di un cluster database, pertanto ci sarà un tempo di inattività di 20-30 secondi, al termine del quale sarà possibile ricominciare a utilizzare il cluster o i cluster database. Per visualizzare o modificare le impostazioni della finestra di manutenzione, accedi alla console Neptune.

In caso di domande o dubbi, il team del supporto AWS è disponibile nei forum della community e tramite il Supporto AWS Premium.

Miglioramenti

- È stato risolto un problema a causa del quale veniva emesso un numero elevato di richieste di caricamento in blocco in rapida successione generando un errore.
- È stato risolto un problema relativo alla dipendenza dai dati a causa del quale a seguito di una query poteva verificarsi un errore interno del server. L'esempio seguente mostra il tipo di query interessata.

  ```
  g.V("my-id123").as("start").outE("knows").has("edgePropertyKey1",
   P.gt(0)).as("myedge").inV()
                .as("end").select("start", "end", "myedge").by("vertexPropertyKey1")
                .by("vertexPropertyKey1").by("edgePropertyKey1")
  ```

- È stato risolto un problema a causa del quale il client Gremlin Java non poteva collegarsi al server utilizzando la stessa connessione WebSocket dopo la scadenza di una query di lunga esecuzione.

- È stato risolto un problema a causa del quale le sequenze di escape incluse come parte della query Gremlin su HTTP o le query basate su stringhe effettuate tramite la connessione WebSocket non erano gestite correttamente.

Introduzione all'uso delle API Amazon Neptune

Le API di gestione di Amazon Neptune supportano gli SDK per la creazione, la gestione e l'eliminazione di istanze e cluster database Neptune, mentre le API dei dati di Neptune supportano gli SDK per il caricamento dei dati nei grafi, l'esecuzione di query, il recupero di informazioni sui dati dei grafi e l'esecuzione di operazioni di machine learning. Queste API sono disponibili in tutti i linguaggi SDK. Firmando automaticamente le richieste delle API, semplificano enormemente l'integrazione di Neptune nelle applicazioni.

Questa pagina fornisce informazioni sulle modalità di utilizzo di queste API.

Azioni IAM con nomi diversi dalle rispettive controparti degli SDK delle API dei dati di Neptune

Quando effettui una chiamata dei metodi delle API Neptune su un cluster con l'autenticazione IAM abilitata, devi avere una policy IAM associata all'utente o al ruolo da cui partono le chiamate che fornisca le autorizzazioni per le azioni che desideri eseguire. Puoi impostare tali autorizzazioni nella policy utilizzando le azioni IAM corrispondenti. Puoi anche limitare le azioni che possono essere eseguire utilizzando le chiavi di condizione IAM.

La maggior parte delle azioni IAM ha lo stesso nome dei metodi API a cui corrispondono, ma alcuni metodi dell'API dei dati hanno nomi diversi, dal momento che alcuni sono condivisi da più metodi. La tabella seguente elenca i metodi dei dati e le azioni IAM corrispondenti:

Nome dell'operazione API dati	Corrispondenze IAM
CancelGremlinQuery (cancel_gremlin_query)	Operazione: `neptune-db: CancelQuery`
CancelLoaderJob (cancel_loader_job)	Operazione: `neptune-db:CancelLoaderJob`
CancelMLDataProcessingJob (cancel_ml_data_processing_job)	Operazione: `neptune-db:CancelMLDataProcessingJob`

Nome dell'operazione API dati	Corrispondenze IAM
CancelMLModelTrainingJob (cancel_ml_model_training_job)	Operazione: `neptune-db:CancelMLModelTrainingJob`
CancelOpenCypherQuery (cancel_open_cypher_query)	Operazione: `neptune-db:` **`CancelQuery`**
CreateMLEndpoint (create_ml_endpoint)	Operazione: `neptune-db:CreateMLEndpoint`
DeleteMLEndpoint (delete_ml_endpoint)	Operazione: `neptune-db:DeleteMLEndpoint`
DeletePropertygraphStatistics (delete_propertygraph_statistics)	Operazione: `neptune-db:` **`DeleteStatistics`**
DeleteSparqlStatistics (delete_sparql_statistics)	Operazione: `neptune-db:` **`DeleteStatistics`**
ExecuteFastReset execute_fast_reset()	Operazione: `neptune-db:` **`ResetDatabase`**
ExecuteGremlinExplainQuery (execute_gremlin_explain_query)	Operazioni: • `neptune-db:` **`ReadDataViaQuery`** • `neptune-db:` **`WriteDataViaQuery`** • `neptune-db:` **`DeleteDataViaQuery`** Chiave di condizione: `neptune-db:QueryLanguage:Gremlin`

Nome dell'operazione API dati	Corrispondenze IAM	
ExecuteGremlinProfileQuery (execute_gremlin_profile_query)	Operazione: `neptune-db:` **`ReadDataViaQuery`** Chiave di condizione: `neptune-db:QueryLanguage:Gremlin`	
ExecuteGremlinQuery (execute_gremlin_query)	Operazioni: • `neptune-db:` **`ReadDataViaQuery`** • `neptune-db:` **`WriteDataViaQuery`** • `neptune-db:` **`DeleteDataViaQuery`** Chiave di condizione: `neptune-db:QueryLanguage:Gremlin`	
ExecuteOpenCypherExplainQuery (execute_open_cypher_explain_query)	Operazione: `neptune-db:` **`ReadDataViaQuery`** Chiave di condizione: `neptune-db:QueryLanguage:OpenCypher`	

Nome dell'operazione API dati	Corrispondenze IAM
ExecuteOpenCypherQuery (execute_open_cypher_query)	Operazioni: • `neptune-db: **ReadDataViaQuery**` • `neptune-db: **WriteDataViaQuery**` • `neptune-db: **DeleteDataViaQuery**` Chiave di condizione: `neptune-db:QueryLanguage:OpenCypher`
GetEngineStatus (get_engine_status)	Operazione: `neptune-db:GetEngineStatus`
GetGremlinQueryStatus (get_gremlin_query_status)	Operazione: `neptune-db: **:GetQueryStatus**` Chiave di condizione: `neptune-db:QueryLanguage:Gremlin`
GetLoaderJobStatus (get_loader_job_status)	Operazione: `neptune-db:GetLoaderJobStatus`
GetMLDataProcessingJob (get_ml_data_processing_job)	Operazione: `neptune-db: **GetMLDataProcessingJobStatus**`
GetMLEndpoint (get_ml_endpoint)	Operazione: `neptune-db: **GetMLEndpointStatus**`

Nome dell'operazione API dati	Corrispondenze IAM	
GetMLModelTrainingJob (get_ml_model_training_job)	Operazione: `neptune-db: GetMLModelTrainingJobStatus`	
GetMLModelTransformJob (get_ml_model_transform_job)	Operazione: `neptune-db: GetMLModelTransformJobStatus`	
GetOpenCypherQueryStatus (get_open_cypher_query_status)	Operazione: `neptune-db: :GetQueryStatus` Chiave di condizione: `neptune-db:QueryLanguage:OpenCypher`	
GetPropertygraphStatistics (get_propertygraph_statistics)	Operazione: `neptune-db: GetStatisticsStatus`	
GetPropertygraphStream (get_propertygraph_stream)	Operazione: `neptune-db: GetStreamRecords` Chiavi di condizione: • `neptune-db:QueryLanguage:Gremlin` • `neptune-db:QueryLanguage:OpenCypher`	
GetPropertygraphSummary (get_propertygraph_summary)	Operazione: `neptune-db: GetGraphSummary`	
GetRDFGraphSummary (get_rdf_graph_summary)	Operazione: `neptune-db: GetGraphSummary`	

Nome dell'operazione API dati	Corrispondenze IAM	
GetSparqlStatistics (get_sparql_statistics)	Operazione: `neptune-db: GetStatisticsStatus`	
GetSparqlStream (get_sparql_stream)	Operazione: `neptune-db: :GetStreamRecords` Chiave di condizione: `neptune-db:QueryLanguage:Sparql`	
ListGremlinQueries (list_gremlin_queries)	Operazione: `neptune-db: :GetQueryStatus` Chiave di condizione: `neptune-db:QueryLanguage:Gremlin`	
ListMLEndpoints (list_ml_endpoints)	Operazione: `neptune-db:ListMLEndpoints`	
ListMLModelTrainingJobs (list_ml_model_training_jobs)	Operazione: `neptune-db:ListMLModelTrainingJobs`	
ListMLModelTransformJobs (list_ml_model_transform_jobs)	Operazione: `neptune-db:ListMLModelTransformJobs`	
ListOpenCypherQueries (list_open_cypher_queries)	Operazione: `neptune-db: :GetQueryStatus` Chiave di condizione: `neptune-db:QueryLanguage:OpenCypher`	

Nome dell'operazione API dati	Corrispondenze IAM
ManagePropertygraphStatistics (manage_propertygraph_statistics)	Operazione: `neptune-db:` **`ManageStatistics`**
ManageSparqlStatistics (manage_sparql_statistics)	Operazione: `neptune-db:` **`ManageStatistics`**
StartLoaderJob (start_loader_job)	Operazione: `neptune-db:StartLoaderJob`
StartMLModelDataProcessingJob (start_ml_data_processing_job)	Operazione: `neptune-db:StartMLModelDataProcessingJob`
StartMLModelTrainingJob (start_ml_model_training_job)	Operazione: `neptune-db:StartMLModelTrainingJob`
StartMLModelTransformJob (start_ml_model_transform_job)	Operazione: `neptune-db:StartMLModelTransformJob`

Documentazione di riferimento delle API di gestione Amazon Neptune

In questo capitolo viene fornita la documentazione delle operazioni delle API Neptune che è possibile utilizzare per gestire e mantenere il proprio cluster database Neptune.

Neptune opera su cluster di server di database che sono collegati in una topologia di replica. Di conseguenza, la gestione di Neptune spesso implica l'implementazione di modifiche in più server e la necessità di accertarsi che tutte le repliche Neptune siano in grado di stare al passo con il server principale.

Poiché Neptune dimensiona in modo trasparente lo storage sottostante in base alla crescita dei dati, la gestione di Neptune richiede relativamente poca gestione dello storage su disco. Analogamente, dato che Neptune esegue backup continui in automatico, un cluster Neptune non richiede pianificazione o tempi di inattività per l'esecuzione dei backup.

Indice

- PendingCloudwatchLogsExports (struttura)
- ClusterPendingModifiedValues (struttura)

- API del database globale Neptune
  - CreateGlobalCluster (azione)
  - DeleteGlobalCluster (azione)
  - ModifyGlobalCluster (azione)
  - DescribeGlobalClusters (azione)
  - FailoverGlobalCluster (azione)
  - RemoveFromGlobalCluster (azione)
  - Strutture:
  - GlobalCluster (struttura)
  - GlobalClusterMember (struttura)
- API per le istanze Neptune
  - CreateDBInstance (operazione)
  - DeleteDBInstance (operazione)
  - ModifyDBInstance (operazione)
  - RebootDBInstance (operazione)
  - DescribeDBInstances (operazione)
  - DescribeOrderableDBInstanceOptions (operazione)
  - DescribeValidDBInstanceModifications (operazione)
  - Strutture:
  - DBInstance (struttura)
  - DBInstanceStatusInfo (struttura)
  - OrderableDBInstanceOption (struttura)
  - PendingModifiedValues (struttura)
  - ValidStorageOptions (struttura)
  - ValidDBInstanceModificationsMessage (struttura)
- API dei parametri Neptune
  - CopyDBParameterGroup (operazione)
  - CopyDBClusterParameterGroup (operazione)

- CreateDBParameterGroup (operazione)
- CreateDBClusterParameterGroup (operazione)
- DeleteDBParameterGroup (operazione)
- DeleteDBClusterParameterGroup (operazione)
- ModifyDBParameterGroup (operazione)
- ModifyDBClusterParameterGroup (operazione)
- ResetDBParameterGroup (operazione)
- ResetDBClusterParameterGroup (operazione)
- DescribeDBParameters (operazione)
- DescribeDBParameterGroups (operazione)
- DescribeDBClusterParameters (operazione)
- DescribeDBClusterParameterGroups (operazione)
- DescribeEngineDefaultParameters (operazione)
- DescribeEngineDefaultClusterParameters (operazione)
- Strutture:
- Parametro (struttura)
- DBParameterGroup (struttura)
- DBClusterParameterGroup (struttura)
- DBParameterGroupStatus (struttura)

- API delle sottoreti Neptune
  - CreateDBSubnetGroup (operazione)
  - DeleteDBSubnetGroup (operazione)
  - ModifyDBSubnetGroup (operazione)
  - DescribeDBSubnetGroups (operazione)
  - Strutture:
  - Sottorete (struttura)
  - DBSubnetGroup (struttura)
- API di snapshot Neptune
  - CreateDBClusterSnapshot (operazione)
  - DeleteDBClusterSnapshot (operazione)

- CopyDBClusterSnapshot (operazione)
- ModifyDbClusterSnapshotAttribute (operazione)
- RestoreDBClusterFromSnapshot (operazione)
- RestoreDBClusterToPointInTime (operazione)
- DescribeDBClusterSnapshots (operazione)
- DescribeDBClusterSnapshotAttributes (operazione)
- Strutture:
- DBClusterSnapshot (struttura)
- DBClusterSnapshotAttribute (struttura)
- DBClusterSnapshotAttributesResult (struttura)

- API Events Neptune
  - CreateEventSubscription (operazione)
  - DeleteEventSubscription (operazione)
  - ModifyEventSubscription (operazione)
  - DescribeEventSubscriptions (operazione)
  - AddSourceIdentifierToSubscription (operazione)
  - RemoveSourceIdentifierFromSubscription (operazione)
  - DescribeEvents (operazione)
  - DescribeEventCategories (operazione)
  - Strutture:
  - Evento (struttura)
  - EventCategoriesMap (struttura)
  - EventSubscription (struttura)
- Altre API Neptune
  - AddTagsToResource (operazione)
  - ListTagsForResource (operazione)
  - RemoveTagsFromResource (operazione)
  - ApplyPendingMaintenanceAction (operazione)
  - DescribePendingMaintenanceActions (operazione)
  - DescribeDBEngineVersions (operazione)

- Strutture:
- DBEngineVersion (struttura)
- EngineDefaults (struttura)
- PendingMaintenanceAction (struttura)
- ResourcePendingMaintenanceActions (struttura)
- UpgradeTarget (struttura)
- Tag (struttura)

- Tipi di dati comuni di Neptune
  - AvailabilityZone (struttura)
  - DBSecurityGroupMembership (struttura)
  - DomainMembership (struttura)
  - DoubleRange (struttura)
  - Endpoint (struttura)
  - Filter (struttura)
  - Range (struttura)
  - ServerlessV2ScalingConfiguration (struttura)
  - ServerlessV2ScalingConfigurationInfo (struttura)
  - Timezone (struttura)
  - VpcSecurityGroupMembership (struttura)
- Le eccezioni Neptune specifiche per le API individuali
  - AuthorizationAlreadyExistsFault (struttura)
  - AuthorizationNotFoundFault (struttura)
  - AuthorizationQuotaExceededFault (struttura)
  - CertificateNotFoundFault (struttura)
  - DBClusterAlreadyExistsFault (struttura)
  - DBClusterNotFoundFault (struttura)
  - DBClusterParameterGroupNotFoundFault (struttura)
  - DBClusterQuotaExceededFault (struttura)
  - DBClusterRoleAlreadyExistsFault (struttura)
  - DBClusterRoleNotFoundFault (struttura)

- DBClusterRoleQuotaExceededFault (struttura)
- DBClusterSnapshotAlreadyExistsFault (struttura)
- DBClusterSnapshotNotFoundFault (struttura)
- DBInstanceAlreadyExistsFault (struttura)
- DBInstanceNotFoundFault (struttura)
- DBLogFileNotFoundFault (struttura)
- DBParameterGroupAlreadyExistsFault (struttura)
- DBParameterGroupNotFoundFault (struttura)
- DBParameterGroupQuotaExceededFault (struttura)
- DBSecurityGroupAlreadyExistsFault (struttura)
- DBSecurityGroupNotFoundFault (struttura)
- DBSecurityGroupNotSupportedFault (struttura)
- DBSecurityGroupQuotaExceededFault (struttura)
- DBSnapshotAlreadyExistsFault (struttura)
- DBSnapshotNotFoundFault (struttura)
- DBSubnetGroupAlreadyExistsFault (struttura)
- DBSubnetGroupDoesNotCoverEnoughAZs (struttura)
- DBSubnetGroupNotAllowedFault (struttura)
- DBSubnetGroupNotFoundFault (struttura)
- DBSubnetGroupQuotaExceededFault (struttura)
- DBSubnetQuotaExceededFault (struttura)
- DBUpgradeDependencyFailureFault (struttura)
- DomainNotFoundFault (struttura)
- EventSubscriptionQuotaExceededFault (struttura)
- GlobalClusterAlreadyExistsFault (struttura)
- GlobalClusterNotFoundFault (struttura)
- GlobalClusterQuotaExceededFault (struttura)
- InstanceQuotaExceededFault (struttura)
- InsufficientDBClusterCapacityFault (struttura)
- InsufficientDBInstanceCapacityFault (struttura)

- InsufficientStorageClusterCapacityFault (struttura)
- InvalidDBClusterEndpointStateFault (struttura)
- InvalidDBClusterSnapshotStateFault (struttura)
- InvalidDBClusterStateFault (struttura)
- InvalidDBInstanceStateFault (struttura)
- InvalidDBParameterGroupStateFault (struttura)
- InvalidDBSecurityGroupStateFault (struttura)
- InvalidDBSnapshotStateFault (struttura)
- InvalidDBSubnetGroupFault (struttura)
- InvalidDBSubnetGroupStateFault (struttura)
- InvalidDBSubnetStateFault (struttura)
- InvalidEventSubscriptionStateFault (struttura)
- InvalidGlobalClusterStateFault (struttura)
- InvalidOptionGroupStateFault (struttura)
- InvalidRestoreFault (struttura)
- InvalidSubnet (struttura)
- InvalidVPCNetworkStateFault (struttura)
- KMSKeyNotAccessibleFault (struttura)
- OptionGroupNotFoundFault (struttura)
- PointInTimeRestoreNotEnabledFault (struttura)
- ProvisionedIopsNotAvailableInAZFault (struttura)
- ResourceNotFoundFault (struttura)
- SNSInvalidTopicFault (struttura)
- SNSNoAuthorizationFault (struttura)
- SNSTopicArnNotFoundFault (struttura)
- SharedSnapshotQuotaExceededFault (struttura)
- SnapshotQuotaExceededFault (struttura)
- SourceNotFoundFault (struttura)
- StorageQuotaExceededFault (struttura)
- StorageTypeNotSupportedFault (struttura)

- SubnetAlreadyInUse (struttura)
- SubscriptionAlreadyExistFault (struttura)
- SubscriptionCategoryNotFoundFault (struttura)
- SubscriptionNotFoundFault (struttura)

API per i cluster di database Neptune

Operazioni:

- CreateDBCluster (operazione)
- DeleteDBCluster (operazione)
- ModifyDBCluster (operazione)
- StartDbCluster (operazione)
- StopDBCluster (operazione)
- AddRoleToDBCluster (operazione)
- RemoveRoleFromDBCluster (operazione)
- FailoverDBCluster (operazione)
- PromoteReadReplicaDBCluster (operazione)
- DescribeDBClusters (operazione)

Strutture:

- DBCluster (struttura)
- DBClusterMember (struttura)
- DBClusterRole (struttura)
- CloudwatchLogsExportConfiguration (struttura)
- PendingCloudwatchLogsExports (struttura)
- ClusterPendingModifiedValues (struttura)

CreateDBCluster (operazione)

Il nome AWS CLI per questa API è: `create-db-cluster`.

Crea un nuovo cluster di database Amazon Neptune.

Puoi utilizzare il parametro `ReplicationSourceIdentifier` per creare il cluster di database come replica di lettura di un altro cluster di database o istanza database Amazon Neptune.

Nota che durante la creazione di un nuovo cluster utilizzando direttamente `CreateDBCluster`, la protezione da eliminazione è disabilitata per impostazione predefinita (quando crei un nuovo cluster di produzione nella console, la protezione da eliminazione è abilitata per impostazione predefinita). Puoi eliminare un cluster database solo se il campo `DeletionProtection` è impostato su `false`.

Richiesta

- AvailabilityZones (nella CLI: `--availability-zones`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco delle zone di disponibilità EC2 in cui possono essere create istanze nel cluster di database.
- BackupRetentionPeriod (nella CLI: `--backup-retention-period`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Il numero di giorni durante i quali vengono conservati i backup automatici. È necessario specificare un valore minimo pari a 1.

  Impostazione predefinita: 1

  Vincoli:

  - Il valore deve essere compreso tra 1 e 35
- CopyTagsToSnapshot (nella CLI: `--copy-tags-to-snapshot`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su *`true`*, i tag vengono copiati in qualsiasi snapshot del cluster database che viene creato.
- DatabaseName (nella CLI: `--database-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome per il database, composto da un massimo di 64 caratteri alfanumerici. Se non specifichi un nome, Amazon Neptune non creerà un database nel cluster di database che stai creando.
- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

L'identificatore del cluster di database. Questo parametro è archiviato come stringa in minuscolo.

Vincoli:

- Deve contenere da 1 a 63 lettere, numeri o trattini.
- Il primo carattere deve essere una lettera.
- Non può terminare con un trattino o contenere due trattini consecutivi.

Esempio: `my-cluster1`

- DBClusterParameterGroupName (nella CLI: `--db-cluster-parameter-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del gruppo di parametri del cluster di database che desideri associare al cluster di database. Se questo argomento viene omesso, viene utilizzato il valore predefinito.

  Vincoli:

  - Se viene specificato, deve corrispondere al nome di un oggetto DBClusterParameterGroup esistente.
- DBSubnetGroupName (nella CLI: `--db-subnet-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Gruppo di sottoreti database da associare al cluster di database.

  Vincoli: deve corrispondere al nome di un oggetto DBSubnetGroup esistente. Non deve essere predefinito.

  Esempio: `mySubnetgroup`
- DeletionProtection (nella CLI: `--deletion-protection`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se per il cluster di database è abilitata la protezione dall'eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione. Per impostazione predefinita, la protezione da eliminazione è abilitata.
- EnableCloudwatchLogsExports (nella CLI: `--enable-cloudwatch-logs-exports`): una stringa di tipo `string` (una stringa con codifica UTF-8).

Elenco di tipi di log che questo cluster database deve esportare in CloudWatch Logs. I tipi di log validi sono: `audit` (per pubblicare log di audit) e `slowquery` (per pubblicare log di slow query). Consulta Pubblicazione di log Neptune nei file di log Amazon CloudWatch.

- EnableIAMDatabaseAuthentication (nella CLI: `--enable-iam-database-authentication`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `true`, abilita l'autenticazione Amazon Identity and Access Management (IAM) per l'intero cluster database (non può essere impostato a livello di istanza).

  Default: `false`.

- Engine (nella CLI: `--engine`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del motore di database da utilizzare per il cluster di database.

  Valori validi: `neptune`

- EngineVersion (nella CLI: `--engine-version`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il numero della versione del motore di database da utilizzare per il nuovo cluster di database.

  Esempio: `1.2.1.0`

- GlobalClusterIdentifier (nella CLI: `--global-cluster-identifier`): un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  L'ID del database globale Neptune a cui aggiungere questo nuovo cluster database.

- KmsKeyId (nella CLI: `--kms-key-id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore della chiave Amazon KMS per un cluster database crittografato.

  L'identificatore della chiave KMS è l'Amazon Resource Name (ARN) per la chiave di crittografia KMS. Se stai creando un cluster database con lo stesso account Amazon che possiede la chiave di crittografia KMS utilizzata per crittografare il nuovo cluster database, puoi utilizzare l'alias della chiave KMS al posto dell'ARN per la chiave di crittografia KMS.

  Se non è specificata una chiave di crittografia in `KmsKeyId`:

- Se `ReplicationSourceIdentifier` identifica un'origine crittografata, Amazon Neptune utilizzerà la chiave di crittografia utilizzata per crittografare l'origine. In caso contrario, Amazon Neptune utilizzerà la chiave di crittografia predefinita.
- Se il parametro `StorageEncrypted` è true e l'oggetto `ReplicationSourceIdentifier` non è specificato, Amazon Neptune utilizzerà la chiave di crittografia predefinita.

Amazon KMS crea la chiave di crittografia predefinita per l'account Amazon. L'account Amazon ha una chiave crittografica predefinita diversa per ogni regione Amazon.

Se crei una replica di lettura di un cluster database crittografato in un'altra regione Amazon, devi impostare `KmsKeyId` su un ID di chiave KMS valido nella regione Amazon di destinazione. Questa chiave viene utilizzata per crittografare la replica di lettura nella regione Amazon.

- Port (nella CLI: `--port`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero della porta su cui le istanze nel cluster di database accettano le connessioni.

  Impostazione predefinita: `8182`
- PreferredBackupWindow (nella CLI: `--preferred-backup-window`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Intervallo di tempo giornaliero durante il quale vengono creati i backup automatici se sono abilitati tramite il parametro `BackupRetentionPeriod`.

  Il valore predefinito è una finestra di 30 minuti selezionata in modo casuale in un blocco di 8 ore per ogni regione Amazon. Per informazioni sui blocchi di tempo disponibili, consulta Finestra di manutenzione Neptune nella Guida per l'utente di Amazon Neptune.

  Vincoli:

  - Il valore deve essere nel formato `hh24:mi-hh24:mi`.
  - Il valore deve essere nel fuso orario UTC (Universal Coordinated Time).
  - Il valore non deve essere in conflitto con la finestra di manutenzione preferita.
  - Il valore deve essere almeno di 30 minuti.
- PreferredMaintenanceWindow (nella CLI: `--preferred-maintenance-window`): una stringa di tipo `string` (una stringa con codifica UTF-8).

Intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.

Formato: `ddd:hh24:mi-ddd:hh24:mi`

Il valore predefinito è una finestra di 30 minuti selezionata in modo casuale in un blocco di 8 ore per ogni regione Amazon, in un giorno casuale della settimana. Per informazioni sui blocchi di tempo disponibili, consulta Finestra di manutenzione Neptune nella Guida per l'utente di Amazon Neptune.

Giorni validi: lunedì, martedì, mercoledì, giovedì, venerdì, sabato, domenica.

Vincoli: finestra di un minimo di 30 minuti.

- PreSignedUrl (nella CLI: `--pre-signed-url`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Questo parametro non è attualmente supportato.
- ReplicationSourceIdentifier (nella CLI: `--replication-source-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) dell'istanza database di origine o del cluster di database, se questo viene creato come replica di lettura.
- ServerlessV2ScalingConfiguration (nella CLI: `--serverless-v2-scaling-configuration`): un oggetto ServerlessV2ScalingConfiguration.

  Contiene la configurazione di dimensionamento di un cluster database Neptune Serverless.

  Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.
- StorageEncrypted (nella CLI: `--storage-encrypted`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database è crittografato.
- StorageType (nella CLI: `--storage-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Tipo di archiviazione per il nuovo cluster di database.

Valori validi:

- **`standard`**: (impostazione predefinita) consente di configurare archiviazione di database dai costi contenuti per applicazioni con un numero di operazioni di I/O da moderato a basso. Se impostato su `standard`, il tipo di archiviazione non viene restituito nella risposta.
- **`iopt1`**: abilita l'archiviazione ottimizzata per l'I/O che è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O che richiedono prezzi prevedibili con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

  L'archiviazione ottimizzata per l'I/O di Neptune è disponibile solo a partire dal rilascio 1.3.0.0 del motore.

- Tags (nella CLI: `--tags`): un array di oggetti Tag.

  Tag da assegnare al nuovo cluster di database.

- VpcSecurityGroupIds (nella CLI: `--vpc-security-group-ids`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco dei gruppi di sicurezza VPC EC2 da associare al cluster di database.

Risposta

Contiene i dettagli di un cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'the section called “DescribeDBClusters”.

- AllocatedStorage: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  `AllocatedStorage` restituisce sempre 1, perché le dimensioni di storage dei cluster di database Neptune non sono fisse, ma vengono regolate automaticamente in base alle esigenze.

- AssociatedRoles: una matrice di oggetti DBClusterRole.

  Fornisce un elenco dei ruoli Amazon Identity and Access Management (IAM) associati al cluster di database. I ruoli IAM associati a un cluster di database concedono l'autorizzazione per l'accesso del cluster di database ad altri servizi Amazon per tuo conto.

- AutomaticRestartTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

Ora in cui il cluster database verrà riavviato automaticamente.

- AvailabilityZones: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 in cui è possibile creare istanze nel cluster di database.

- BacktrackConsumedChangeRecords: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.

- BacktrackWindow: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.

- BackupRetentionPeriod: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.

- Capacity: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Non supportato da Neptune.

- CloneGroupId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identifica il gruppo di cloni a cui è associato il cluster di database.

- ClusterCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).

- CopyTagsToSnapshot: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su *`true`*, i tag vengono copiati in qualsiasi snapshot del cluster database che viene creato.

- CrossAccountClone: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `true`, il cluster database può essere clonato su più account.

- DatabaseName: una stringa di tipo `string` (una stringa con codifica UTF-8).

Contiene il nome del database iniziale di questo cluster di database, fornito al momento della creazione, se è stato specificato un nome quando il cluster di database è stato creato. Questo stesso nome viene restituito per la durata del cluster di database.

- DBClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per il cluster di database.

- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore del cluster di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un cluster di database.

- DBClusterMembers: una matrice di oggetti DBClusterMember.

  Fornisce l'elenco delle istanze che compongono il cluster di database.

- DBClusterParameterGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del gruppo di parametri del cluster di database per il cluster stesso.

- DbClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore univoco e non modificabile della regione Amazon per il cluster database. Questo identificativo è disponibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per il cluster database.

- DBSubnetGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica le informazioni sul gruppo di sottoreti associato al cluster di database, tra cui nome, descrizione e sottoreti nel gruppo.

- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se per il cluster di database è abilitata o meno la protezione da eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione.

- EarliestBacktrackTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Non supportato da Neptune.

- EarliestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica il primo orario possibile per il ripristino point-in-time di un database.

- EnabledCloudwatchLogsExports: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questo cluster database può esportare in CloudWatch Logs. I tipi di log validi sono: `audit` (per pubblicare log di audit su CloudWatch) e slowquery (per pubblicare log di slow query su CloudWatch). Consulta Pubblicazione di log Neptune nei file di log Amazon CloudWatch.

- Endpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'endpoint di connessione per l'istanza primaria del cluster di database.

- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questo cluster di database.

- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.

- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.

- HostedZoneId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'ID che Amazon Route 53 assegna al momento della creazione di una zona ospitata.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.

- IOOptimizedNextAllowedModificationTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La prossima volta puoi modificare il cluster di database per utilizzare il tipo di archiviazione `iopt1`.

- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per il cluster di database crittografato.

- LatestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.

- MultiAZ: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database ha istanze in più zone di disponibilità.

- PendingModifiedValues: un oggetto ClusterPendingModifiedValues.

  Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione `ModifyDBCluster` e contiene le modifiche che verranno applicate nella finestra di manutenzione successiva.

- PercentProgress: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'avanzamento dell'operazione sotto forma di percentuale.

- Port: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto il motore di database.

- PreferredBackupWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.

- ReaderEndpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'endpoint di lettura per il cluster di database. L'endpoint di lettura per un cluster di database bilancia il carico delle connessioni tra le repliche di lettura disponibili in un cluster di database. Man mano che i client richiedono nuove connessioni all'endpoint di lettura, Neptune distribuisce tali richieste fra le repliche di lettura nel cluster di database. Questa funzionalità è utile per bilanciare il carico di lavoro di lettura tra più repliche di lettura nel cluster di database.

  Se si verifica un failover e la replica di lettura a cui sei connesso viene promossa a istanza primaria, la connessione viene interrotta. Per continuare a inviare il carico di lavoro di lettura ad altre repliche di lettura nel cluster, puoi quindi riconnetterti all'endpoint di lettura.

- ReadReplicaIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

Contiene uno o più identificatori delle repliche di lettura associate a questo cluster di database.

- ReplicationSourceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ReplicationType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ServerlessV2ScalingConfiguration: un oggetto ServerlessV2ScalingConfigurationInfo.

  Mostra la configurazione del dimensionamento per un cluster database Neptune Serverless.

  Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del cluster di database.

- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database è crittografato.

- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione utilizzato dal cluster di database.

  Valori validi:

  - **`standard`**: (impostazione predefinita) fornisce archiviazione di database dai costi contenuti per applicazioni con un numero di operazioni di I/O da moderato a basso.
  - **`iopt1`**: abilita l'archiviazione ottimizzata per l'I/O che è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O che richiedono prezzi prevedibili con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

    L'archiviazione ottimizzata per l'I/O di Neptune è disponibile solo a partire dal rilascio 1.3.0.0 del motore.

- VpcSecurityGroups: una matrice di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco dei gruppi di sicurezza VPC a cui appartiene il cluster di database.

Errori

- DBClusterAlreadyExistsFault
- InsufficientStorageClusterCapacityFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- InvalidVPCNetworkStateFault
- InvalidDBClusterStateFault
- InvalidDBSubnetGroupStateFault
- InvalidSubnet
- InvalidDBInstanceStateFault
- DBClusterParameterGroupNotFoundFault
- KMSKeyNotAccessibleFault
- DBClusterNotFoundFault
- DBInstanceNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

DeleteDBCluster (operazione)

Il nome AWS CLI per questa API è: `delete-db-cluster`.

L'operazione DeleteDBCluster elimina un cluster di database di cui è stato effettuato il provisioning in precedenza. Quando elimini un cluster di database, tutti i backup automatici per tale cluster di database vengono eliminati e non possono essere recuperati. Gli snapshot dei cluster di database manuali per il cluster di database specificato non vengono eliminati.

Nota che se è abilitata la protezione da eliminazione, il cluster database non può essere eliminato. Per eliminarlo, devi prima impostare il suo campo `DeletionProtection` su `False`.

Richiesta

- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore del cluster di database da eliminare. Questo parametro non fa distinzione tra maiuscole e minuscole.

  Vincoli:

  - Deve corrispondere a un oggetto DBClusterIdentifier esistente.
- FinalDBSnapshotIdentifier (nella CLI: `--final-db-snapshot-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore dello snapshot del cluster di database per il nuovo snapshot creato quando l'impostazione di `SkipFinalSnapshot` è `false`.

  > **Note**
  >
  > Se specifichi questo parametro e imposti anche il parametro `SkipFinalShapshot` su true, viene generato un errore.

  Vincoli:

  - Deve contenere da 1 a 255 lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera
  - Non può terminare con un trattino o contenere due trattini consecutivi
- SkipFinalSnapshot (nella CLI: `--skip-final-snapshot`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Determina se viene creato uno snapshot del cluster di database finale prima dell'eliminazione del cluster di database. Se viene specificato `true`, non viene creato alcuno snapshot del cluster di database. Se viene specificato `false`, viene creato uno snapshot del cluster di database prima dell'eliminazione del cluster di database.

  > **Note**
  >
  > È necessario specificare un parametro `FinalDBSnapshotIdentifier` se `SkipFinalSnapshot` è `false`.

Impostazione predefinita: `false`

Risposta

Contiene i dettagli di un cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'the section called “DescribeDBClusters”.

- AllocatedStorage: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  `AllocatedStorage` restituisce sempre 1, perché le dimensioni di storage dei cluster di database Neptune non sono fisse, ma vengono regolate automaticamente in base alle esigenze.
- AssociatedRoles: una matrice di oggetti DBClusterRole.

  Fornisce un elenco dei ruoli Amazon Identity and Access Management (IAM) associati al cluster di database. I ruoli IAM associati a un cluster di database concedono l'autorizzazione per l'accesso del cluster di database ad altri servizi Amazon per tuo conto.
- AutomaticRestartTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Ora in cui il cluster database verrà riavviato automaticamente.
- AvailabilityZones: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 in cui è possibile creare istanze nel cluster di database.
- BacktrackConsumedChangeRecords: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.
- BacktrackWindow: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.
- BackupRetentionPeriod: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.

- Capacity: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Non supportato da Neptune.
- CloneGroupId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identifica il gruppo di cloni a cui è associato il cluster di database.
- ClusterCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).
- CopyTagsToSnapshot: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su *`true`*, i tag vengono copiati in qualsiasi snapshot del cluster database che viene creato.
- CrossAccountClone: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `true`, il cluster database può essere clonato su più account.
- DatabaseName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nome del database iniziale di questo cluster di database, fornito al momento della creazione, se è stato specificato un nome quando il cluster di database è stato creato. Questo stesso nome viene restituito per la durata del cluster di database.
- DBClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per il cluster di database.
- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore del cluster di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un cluster di database.
- DBClusterMembers: una matrice di oggetti DBClusterMember.

  Fornisce l'elenco delle istanze che compongono il cluster di database.
- DBClusterParameterGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del gruppo di parametri del cluster di database per il cluster stesso.

- DbClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore univoco e non modificabile della regione Amazon per il cluster database. Questo identificativo è disponibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per il cluster database.

- DBSubnetGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica le informazioni sul gruppo di sottoreti associato al cluster di database, tra cui nome, descrizione e sottoreti nel gruppo.

- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se per il cluster di database è abilitata o meno la protezione da eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione.

- EarliestBacktrackTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Non supportato da Neptune.

- EarliestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica il primo orario possibile per il ripristino point-in-time di un database.

- EnabledCloudwatchLogsExports: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questo cluster database può esportare in CloudWatch Logs. I tipi di log validi sono: `audit` (per pubblicare log di audit su CloudWatch) e slowquery (per pubblicare log di slow query su CloudWatch). Consulta Pubblicazione di log Neptune nei file di log Amazon CloudWatch.

- Endpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'endpoint di connessione per l'istanza primaria del cluster di database.

- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questo cluster di database.

- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.

- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.
- HostedZoneId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'ID che Amazon Route 53 assegna al momento della creazione di una zona ospitata.
- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.
- IOOptimizedNextAllowedModificationTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La prossima volta puoi modificare il cluster di database per utilizzare il tipo di archiviazione `iopt1`.
- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per il cluster di database crittografato.
- LatestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.
- MultiAZ: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database ha istanze in più zone di disponibilità.
- PendingModifiedValues: un oggetto ClusterPendingModifiedValues.

  Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione `ModifyDBCluster` e contiene le modifiche che verranno applicate nella finestra di manutenzione successiva.
- PercentProgress: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'avanzamento dell'operazione sotto forma di percentuale.
- Port: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

Specifica la porta su cui è in ascolto il motore di database.

- PreferredBackupWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.

- ReaderEndpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'endpoint di lettura per il cluster di database. L'endpoint di lettura per un cluster di database bilancia il carico delle connessioni tra le repliche di lettura disponibili in un cluster di database. Man mano che i client richiedono nuove connessioni all'endpoint di lettura, Neptune distribuisce tali richieste fra le repliche di lettura nel cluster di database. Questa funzionalità è utile per bilanciare il carico di lavoro di lettura tra più repliche di lettura nel cluster di database.

  Se si verifica un failover e la replica di lettura a cui sei connesso viene promossa a istanza primaria, la connessione viene interrotta. Per continuare a inviare il carico di lavoro di lettura ad altre repliche di lettura nel cluster, puoi quindi riconnetterti all'endpoint di lettura.

- ReadReplicaIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori delle repliche di lettura associate a questo cluster di database.

- ReplicationSourceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ReplicationType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ServerlessV2ScalingConfiguration: un oggetto ServerlessV2ScalingConfigurationInfo.

  Mostra la configurazione del dimensionamento per un cluster database Neptune Serverless.

  Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

Specifica lo stato corrente del cluster di database.

- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database è crittografato.

- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione utilizzato dal cluster di database.

  Valori validi:

  - **`standard`**: (impostazione predefinita) fornisce archiviazione di database dai costi contenuti per applicazioni con un numero di operazioni di I/O da moderato a basso.
  - **`iopt1`**: abilita l'archiviazione ottimizzata per l'I/O che è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O che richiedono prezzi prevedibili con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

    L'archiviazione ottimizzata per l'I/O di Neptune è disponibile solo a partire dal rilascio 1.3.0.0 del motore.

- VpcSecurityGroups: una matrice di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco dei gruppi di sicurezza VPC a cui appartiene il cluster di database.

Errori

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- DBClusterSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault
- InvalidDBClusterSnapshotStateFault

ModifyDBCluster (operazione)

Il nome AWS CLI per questa API è: `modify-db-cluster`.

Modifica un'impostazione per un cluster di database. Puoi modificare uno o più parametri di configurazione del database specificando questi parametri e i nuovi valori nella richiesta.

Richiesta

- AllowMajorVersionUpgrade (nella CLI: `--allow-major-version-upgrade`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica che gli aggiornamenti tra versioni principali diverse sono permessi.

  Vincoli: devi impostare il flag allow-major-version-upgrade quando fornisci un `EngineVersion` parametro che utilizza una versione principale diversa rispetto a quella attuale del cluster database.
- ApplyImmediately (nella CLI: `--apply-immediately`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Valore che specifica se le modifiche in questa richiesta e le eventuali modifiche in sospeso vengono applicate in modo asincrono appena possibile, indipendentemente dall'impostazione di `PreferredMaintenanceWindow` per il cluster di database. Se questo parametro è impostato su `false`, le modifiche al cluster di database vengono applicate durante la finestra di manutenzione successiva.

  Il parametro `ApplyImmediately` influisce solo sui valori di `NewDBClusterIdentifier`. Se imposti il valore del parametro `ApplyImmediately` su false, le modifiche ai valori di `NewDBClusterIdentifier` vengono applicate durante la finestra di manutenzione successiva. Tutte le altre modifiche vengono applicate immediatamente, indipendentemente dal valore del parametro `ApplyImmediately`.

  Impostazione predefinita: `false`
- BackupRetentionPeriod (nella CLI: `--backup-retention-period`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Il numero di giorni durante i quali vengono conservati i backup automatici. È necessario specificare un valore minimo pari a 1.

  Impostazione predefinita: 1

  Vincoli:

  - Il valore deve essere compreso tra 1 e 35
- CloudwatchLogsExportConfiguration (nella CLI: `--cloudwatch-logs-export-configuration`): un oggetto CloudwatchLogsExportConfiguration.

L'impostazione di configurazione per i tipi di log per consentire l'esportazione in CloudWatch Logs per un determinato cluster di database. Vedi Utilizzo della CLI per pubblicare i log di audit di Neptune su CloudWatch Logs.

- CopyTagsToSnapshot (nella CLI: `--copy-tags-to-snapshot`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su *`true`*, i tag vengono copiati in qualsiasi snapshot del cluster database che viene creato.

- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore del cluster di database da modificare. Questo parametro non fa distinzione tra maiuscole e minuscole.

  Vincoli:

  - Deve corrispondere all'identificatore di un oggetto DBCluster esistente.

- DBClusterParameterGroupName (nella CLI: `--db-cluster-parameter-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del gruppo di parametri del cluster di database da utilizzare per il cluster di database.

- DBInstanceParameterGroupName (nella CLI: `--db-instance-parameter-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di parametri del database da applicare a tutte le istanze del cluster di database.

  > **Note**
  >
  > Se applichi un gruppo di parametri utilizzando `DBInstanceParameterGroupName`, le modifiche ai parametri non vengono applicate durante la finestra di manutenzione successiva, ma immediatamente.

  Impostazione predefinita: impostazione del nome esistente

  Vincoli:

  - Il gruppo di parametri del database deve trovarsi nella stessa famiglia del gruppo di parametri della versione del cluster database di destinazione.

  - Il parametro `DBInstanceParameterGroupName` è valido solo in combinazione con il parametro `AllowMajorVersionUpgrade`.
- DeletionProtection (nella CLI: `--deletion-protection`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se per il cluster di database è abilitata la protezione dall'eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione. Per impostazione predefinita, la protezione da eliminazione è disabilitata.
- EnableIAMDatabaseAuthentication (nella CLI: `--enable-iam-database-authentication`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  True per abilitare la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database, in caso contrario false.

  Impostazione predefinita: `false`
- EngineVersion (nella CLI: `--engine-version`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Numero di versione del motore di database a cui eseguire l'aggiornamento. La modifica di questo parametro provoca un'interruzione. La modifica viene applicata durante la finestra di manutenzione successiva, a meno che il parametro `ApplyImmediately` non sia impostato su true.

  Per un elenco di versioni valide del motore, consulta le versioni del motore per Amazon Neptune, oppure chiama the section called “DescribeDBEngineVersions”.
- NewDBClusterIdentifier (nella CLI: `--new-db-cluster-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nuovo identificatore del cluster di database quando un cluster di database viene rinominato. Questo valore è archiviato come stringa in caratteri minuscoli.

  Vincoli:

  - Deve contenere da 1 a 63 lettere, numeri o trattini
  - Il primo carattere deve essere una lettera
  - Non può terminare con un trattino o contenere due trattini consecutivi

  Esempio: `my-cluster2`

- Port (nella CLI: `--port`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Il numero della porta sulla quale il cluster di database accetta connessioni.

  Vincoli: il valore deve essere compreso nell'intervallo `1150-65535`

  Impostazione predefinita: la stessa porta del cluster di database originale.
- PreferredBackupWindow (nella CLI: `--preferred-backup-window`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Intervallo di tempo giornaliero durante il quale vengono creati i backup automatici, se sono abilitati tramite il parametro `BackupRetentionPeriod`.

  Il valore predefinito è una finestra di 30 minuti selezionata in modo casuale in un blocco di 8 ore per ogni regione Amazon.

  Vincoli:

  - Il valore deve essere nel formato `hh24:mi-hh24:mi`.
  - Il valore deve essere nel fuso orario UTC (Universal Coordinated Time).
  - Il valore non deve essere in conflitto con la finestra di manutenzione preferita.
  - Il valore deve essere almeno di 30 minuti.
- PreferredMaintenanceWindow (nella CLI: `--preferred-maintenance-window`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.

  Formato: `ddd:hh24:mi-ddd:hh24:mi`

  Il valore predefinito è una finestra di 30 minuti selezionata in modo casuale in un blocco di 8 ore per ogni regione Amazon, in un giorno casuale della settimana.

  Giorni validi: lunedì, martedì, mercoledì, giovedì, venerdì, sabato, domenica.

  Vincoli: finestra di un minimo di 30 minuti.
- ServerlessV2ScalingConfiguration (nella CLI: `--serverless-v2-scaling-configuration`): un oggetto ServerlessV2ScalingConfiguration.

Contiene la configurazione di dimensionamento di un cluster database Neptune Serverless.

Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

- StorageType (nella CLI: `--storage-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione da associare al cluster di database.

  Valori validi:

  - **`standard`**: (impostazione predefinita) consente di configurare archiviazione di database dai costi contenuti per applicazioni con un numero di operazioni di I/O da moderato a basso.
  - **`iopt1`**: abilita l'archiviazione ottimizzata per l'I/O che è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O che richiedono prezzi prevedibili con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

    L'archiviazione ottimizzata per l'I/O di Neptune è disponibile solo a partire dal rilascio 1.3.0.0 del motore.
- VpcSecurityGroupIds (nella CLI: `--vpc-security-group-ids`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco dei gruppi di sicurezza VPC a cui appartiene il cluster di database.

Risposta

Contiene i dettagli di un cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'the section called “DescribeDBClusters”.

- AllocatedStorage: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  `AllocatedStorage` restituisce sempre 1, perché le dimensioni di storage dei cluster di database Neptune non sono fisse, ma vengono regolate automaticamente in base alle esigenze.
- AssociatedRoles: una matrice di oggetti DBClusterRole.

Fornisce un elenco dei ruoli Amazon Identity and Access Management (IAM) associati al cluster di database. I ruoli IAM associati a un cluster di database concedono l'autorizzazione per l'accesso del cluster di database ad altri servizi Amazon per tuo conto.

- AutomaticRestartTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Ora in cui il cluster database verrà riavviato automaticamente.

- AvailabilityZones: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 in cui è possibile creare istanze nel cluster di database.

- BacktrackConsumedChangeRecords: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.

- BacktrackWindow: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.

- BackupRetentionPeriod: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.

- Capacity: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Non supportato da Neptune.

- CloneGroupId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identifica il gruppo di cloni a cui è associato il cluster di database.

- ClusterCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).

- CopyTagsToSnapshot: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

Se impostato su *true*, i tag vengono copiati in qualsiasi snapshot del cluster database che viene creato.

- CrossAccountClone: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `true`, il cluster database può essere clonato su più account.

- DatabaseName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nome del database iniziale di questo cluster di database, fornito al momento della creazione, se è stato specificato un nome quando il cluster di database è stato creato. Questo stesso nome viene restituito per la durata del cluster di database.

- DBClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per il cluster di database.

- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore del cluster di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un cluster di database.

- DBClusterMembers: una matrice di oggetti DBClusterMember.

  Fornisce l'elenco delle istanze che compongono il cluster di database.

- DBClusterParameterGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del gruppo di parametri del cluster di database per il cluster stesso.

- DbClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore univoco e non modificabile della regione Amazon per il cluster database. Questo identificativo è disponibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per il cluster database.

- DBSubnetGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica le informazioni sul gruppo di sottoreti associato al cluster di database, tra cui nome, descrizione e sottoreti nel gruppo.

- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se per il cluster di database è abilitata o meno la protezione da eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione.

- EarliestBacktrackTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Non supportato da Neptune.

- EarliestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica il primo orario possibile per il ripristino point-in-time di un database.

- EnabledCloudwatchLogsExports: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questo cluster database può esportare in CloudWatch Logs. I tipi di log validi sono: `audit` (per pubblicare log di audit su CloudWatch) e slowquery (per pubblicare log di slow query su CloudWatch). Consulta Pubblicazione di log Neptune nei file di log Amazon CloudWatch.

- Endpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'endpoint di connessione per l'istanza primaria del cluster di database.

- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questo cluster di database.

- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.

- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.

- HostedZoneId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'ID che Amazon Route 53 assegna al momento della creazione di una zona ospitata.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.

- IOOptimizedNextAllowedModificationTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La prossima volta puoi modificare il cluster di database per utilizzare il tipo di archiviazione `iopt1`.
- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per il cluster di database crittografato.
- LatestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.
- MultiAZ: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database ha istanze in più zone di disponibilità.
- PendingModifiedValues: un oggetto ClusterPendingModifiedValues.

  Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione `ModifyDBCluster` e contiene le modifiche che verranno applicate nella finestra di manutenzione successiva.
- PercentProgress: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'avanzamento dell'operazione sotto forma di percentuale.
- Port: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto il motore di database.
- PreferredBackupWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.
- ReaderEndpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'endpoint di lettura per il cluster di database. L'endpoint di lettura per un cluster di database bilancia il carico delle connessioni tra le repliche di lettura disponibili in un cluster di database. Man mano che i client richiedono nuove connessioni all'endpoint di lettura, Neptune distribuisce tali

richieste fra le repliche di lettura nel cluster di database. Questa funzionalità è utile per bilanciare il carico di lavoro di lettura tra più repliche di lettura nel cluster di database.

Se si verifica un failover e la replica di lettura a cui sei connesso viene promossa a istanza primaria, la connessione viene interrotta. Per continuare a inviare il carico di lavoro di lettura ad altre repliche di lettura nel cluster, puoi quindi riconnetterti all'endpoint di lettura.

- ReadReplicaIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori delle repliche di lettura associate a questo cluster di database.

- ReplicationSourceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ReplicationType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ServerlessV2ScalingConfiguration: un oggetto ServerlessV2ScalingConfigurationInfo.

  Mostra la configurazione del dimensionamento per un cluster database Neptune Serverless.

  Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del cluster di database.

- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database è crittografato.

- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione utilizzato dal cluster di database.

  Valori validi:

  - **`standard`**: (impostazione predefinita) fornisce archiviazione di database dai costi contenuti per applicazioni con un numero di operazioni di I/O da moderato a basso.
  - **`iopt1`**: abilita l'archiviazione ottimizzata per l'I/O che è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O che richiedono prezzi prevedibili con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

L'archiviazione ottimizzata per l'I/O di Neptune è disponibile solo a partire dal rilascio 1.3.0.0 del motore.

- VpcSecurityGroups: una matrice di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco dei gruppi di sicurezza VPC a cui appartiene il cluster di database.

Errori

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- InvalidVPCNetworkStateFault
- InvalidDBSubnetGroupStateFault
- InvalidSubnet
- DBClusterParameterGroupNotFoundFault
- InvalidDBSecurityGroupStateFault
- InvalidDBInstanceStateFault
- DBClusterAlreadyExistsFault
- StorageTypeNotSupportedFault

StartDbCluster (operazione)

Il nome AWS CLI per questa API è: `start-db-cluster`.

Avvia un cluster database Amazon Neptune che è stato interrotto utilizzando la console Amazon, il comando stop-db-cluster di Amazon CLI o l'API StopDBCluster.

Richiesta

- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore del cluster di database del cluster di database Neptune da avviare. Questo parametro è archiviato come stringa in minuscolo.

Risposta

Contiene i dettagli di un cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'the section called “DescribeDBClusters”.

- AllocatedStorage: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  `AllocatedStorage` restituisce sempre 1, perché le dimensioni di storage dei cluster di database Neptune non sono fisse, ma vengono regolate automaticamente in base alle esigenze.
- AssociatedRoles: una matrice di oggetti DBClusterRole.

  Fornisce un elenco dei ruoli Amazon Identity and Access Management (IAM) associati al cluster di database. I ruoli IAM associati a un cluster di database concedono l'autorizzazione per l'accesso del cluster di database ad altri servizi Amazon per tuo conto.
- AutomaticRestartTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Ora in cui il cluster database verrà riavviato automaticamente.
- AvailabilityZones: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 in cui è possibile creare istanze nel cluster di database.
- BacktrackConsumedChangeRecords: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.
- BacktrackWindow: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.
- BackupRetentionPeriod: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.
- Capacity: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Non supportato da Neptune.

- CloneGroupId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identifica il gruppo di cloni a cui è associato il cluster di database.

- ClusterCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).

- CopyTagsToSnapshot: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su *`true`*, i tag vengono copiati in qualsiasi snapshot del cluster database che viene creato.

- CrossAccountClone: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `true`, il cluster database può essere clonato su più account.

- DatabaseName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nome del database iniziale di questo cluster di database, fornito al momento della creazione, se è stato specificato un nome quando il cluster di database è stato creato. Questo stesso nome viene restituito per la durata del cluster di database.

- DBClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per il cluster di database.

- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore del cluster di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un cluster di database.

- DBClusterMembers: una matrice di oggetti DBClusterMember.

  Fornisce l'elenco delle istanze che compongono il cluster di database.

- DBClusterParameterGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del gruppo di parametri del cluster di database per il cluster stesso.

- DbClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

Identificatore univoco e non modificabile della regione Amazon per il cluster database. Questo identificativo è disponibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per il cluster database.

- DBSubnetGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica le informazioni sul gruppo di sottoreti associato al cluster di database, tra cui nome, descrizione e sottoreti nel gruppo.
- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se per il cluster di database è abilitata o meno la protezione da eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione.
- EarliestBacktrackTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Non supportato da Neptune.
- EarliestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica il primo orario possibile per il ripristino point-in-time di un database.
- EnabledCloudwatchLogsExports: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questo cluster database può esportare in CloudWatch Logs. I tipi di log validi sono: `audit` (per pubblicare log di audit su CloudWatch) e slowquery (per pubblicare log di slow query su CloudWatch). Consulta Pubblicazione di log Neptune nei file di log Amazon CloudWatch.
- Endpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'endpoint di connessione per l'istanza primaria del cluster di database.
- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questo cluster di database.
- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.
- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.

- HostedZoneId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'ID che Amazon Route 53 assegna al momento della creazione di una zona ospitata.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.

- IOOptimizedNextAllowedModificationTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La prossima volta puoi modificare il cluster di database per utilizzare il tipo di archiviazione `iopt1`.

- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per il cluster di database crittografato.

- LatestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.

- MultiAZ: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database ha istanze in più zone di disponibilità.

- PendingModifiedValues: un oggetto ClusterPendingModifiedValues.

  Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione `ModifyDBCluster` e contiene le modifiche che verranno applicate nella finestra di manutenzione successiva.

- PercentProgress: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'avanzamento dell'operazione sotto forma di percentuale.

- Port: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto il motore di database.

- PreferredBackupWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.

- ReaderEndpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'endpoint di lettura per il cluster di database. L'endpoint di lettura per un cluster di database bilancia il carico delle connessioni tra le repliche di lettura disponibili in un cluster di database. Man mano che i client richiedono nuove connessioni all'endpoint di lettura, Neptune distribuisce tali richieste fra le repliche di lettura nel cluster di database. Questa funzionalità è utile per bilanciare il carico di lavoro di lettura tra più repliche di lettura nel cluster di database.

  Se si verifica un failover e la replica di lettura a cui sei connesso viene promossa a istanza primaria, la connessione viene interrotta. Per continuare a inviare il carico di lavoro di lettura ad altre repliche di lettura nel cluster, puoi quindi riconnetterti all'endpoint di lettura.

- ReadReplicaIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori delle repliche di lettura associate a questo cluster di database.

- ReplicationSourceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ReplicationType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ServerlessV2ScalingConfiguration: un oggetto ServerlessV2ScalingConfigurationInfo.

  Mostra la configurazione del dimensionamento per un cluster database Neptune Serverless.

  Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del cluster di database.

- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

Specifica se il cluster di database è crittografato.

- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione utilizzato dal cluster di database.

  Valori validi:

  - **`standard`**: (impostazione predefinita) fornisce archiviazione di database dai costi contenuti per applicazioni con un numero di operazioni di I/O da moderato a basso.
  - **`iopt1`**: abilita l'archiviazione ottimizzata per l'I/O che è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O che richiedono prezzi prevedibili con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

    L'archiviazione ottimizzata per l'I/O di Neptune è disponibile solo a partire dal rilascio 1.3.0.0 del motore.
- VpcSecurityGroups: una matrice di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco dei gruppi di sicurezza VPC a cui appartiene il cluster di database.

Errori

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- InvalidDBInstanceStateFault

StopDBCluster (operazione)

Il nome AWS CLI per questa API è: `stop-db-cluster`.

Arresta un cluster di database Amazon Neptune. Quando si arresta un cluster di database, Neptune conserva i metadati del cluster di database, inclusi gli endpoint e i gruppi di parametri database.

Neptune conserva inoltre i log delle transazioni in modo da poter eseguire un ripristino point-in-time, se necessario.

Richiesta

- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore del cluster di database del cluster di database Neptune da arrestare. Questo parametro è archiviato come stringa in minuscolo.

Risposta

Contiene i dettagli di un cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'the section called “DescribeDBClusters”.

- AllocatedStorage: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  `AllocatedStorage` restituisce sempre 1, perché le dimensioni di storage dei cluster di database Neptune non sono fisse, ma vengono regolate automaticamente in base alle esigenze.
- AssociatedRoles: una matrice di oggetti DBClusterRole.

  Fornisce un elenco dei ruoli Amazon Identity and Access Management (IAM) associati al cluster di database. I ruoli IAM associati a un cluster di database concedono l'autorizzazione per l'accesso del cluster di database ad altri servizi Amazon per tuo conto.
- AutomaticRestartTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Ora in cui il cluster database verrà riavviato automaticamente.
- AvailabilityZones: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 in cui è possibile creare istanze nel cluster di database.
- BacktrackConsumedChangeRecords: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.
- BacktrackWindow: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.

- BackupRetentionPeriod: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.

- Capacity: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Non supportato da Neptune.

- CloneGroupId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identifica il gruppo di cloni a cui è associato il cluster di database.

- ClusterCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).

- CopyTagsToSnapshot: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su *`true`*, i tag vengono copiati in qualsiasi snapshot del cluster database che viene creato.

- CrossAccountClone: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `true`, il cluster database può essere clonato su più account.

- DatabaseName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nome del database iniziale di questo cluster di database, fornito al momento della creazione, se è stato specificato un nome quando il cluster di database è stato creato. Questo stesso nome viene restituito per la durata del cluster di database.

- DBClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per il cluster di database.

- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore del cluster di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un cluster di database.

- DBClusterMembers: una matrice di oggetti DBClusterMember.

Fornisce l'elenco delle istanze che compongono il cluster di database.

- DBClusterParameterGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del gruppo di parametri del cluster di database per il cluster stesso.

- DbClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore univoco e non modificabile della regione Amazon per il cluster database. Questo identificativo è disponibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per il cluster database.

- DBSubnetGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica le informazioni sul gruppo di sottoreti associato al cluster di database, tra cui nome, descrizione e sottoreti nel gruppo.

- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se per il cluster di database è abilitata o meno la protezione da eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione.

- EarliestBacktrackTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Non supportato da Neptune.

- EarliestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica il primo orario possibile per il ripristino point-in-time di un database.

- EnabledCloudwatchLogsExports: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questo cluster database può esportare in CloudWatch Logs. I tipi di log validi sono: `audit` (per pubblicare log di audit su CloudWatch) e slowquery (per pubblicare log di slow query su CloudWatch). Consulta Pubblicazione di log Neptune nei file di log Amazon CloudWatch.

- Endpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'endpoint di connessione per l'istanza primaria del cluster di database.

- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questo cluster di database.

- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.

- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.

- HostedZoneId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'ID che Amazon Route 53 assegna al momento della creazione di una zona ospitata.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.

- IOOptimizedNextAllowedModificationTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La prossima volta puoi modificare il cluster di database per utilizzare il tipo di archiviazione `iopt1`.

- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per il cluster di database crittografato.

- LatestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.

- MultiAZ: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database ha istanze in più zone di disponibilità.

- PendingModifiedValues: un oggetto ClusterPendingModifiedValues.

  Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione `ModifyDBCluster` e contiene le modifiche che verranno applicate nella finestra di manutenzione successiva.

- PercentProgress: una stringa di tipo `string` (una stringa con codifica UTF-8).

Specifica l'avanzamento dell'operazione sotto forma di percentuale.

- Port: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto il motore di database.

- PreferredBackupWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.

- ReaderEndpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'endpoint di lettura per il cluster di database. L'endpoint di lettura per un cluster di database bilancia il carico delle connessioni tra le repliche di lettura disponibili in un cluster di database. Man mano che i client richiedono nuove connessioni all'endpoint di lettura, Neptune distribuisce tali richieste fra le repliche di lettura nel cluster di database. Questa funzionalità è utile per bilanciare il carico di lavoro di lettura tra più repliche di lettura nel cluster di database.

  Se si verifica un failover e la replica di lettura a cui sei connesso viene promossa a istanza primaria, la connessione viene interrotta. Per continuare a inviare il carico di lavoro di lettura ad altre repliche di lettura nel cluster, puoi quindi riconnetterti all'endpoint di lettura.

- ReadReplicaIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori delle repliche di lettura associate a questo cluster di database.

- ReplicationSourceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ReplicationType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ServerlessV2ScalingConfiguration: un oggetto ServerlessV2ScalingConfigurationInfo.

  Mostra la configurazione del dimensionamento per un cluster database Neptune Serverless.

Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del cluster di database.

- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database è crittografato.

- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione utilizzato dal cluster di database.

  Valori validi:

  - **`standard`**: (impostazione predefinita) fornisce archiviazione di database dai costi contenuti per applicazioni con un numero di operazioni di I/O da moderato a basso.
  - **`iopt1`**: abilita l'archiviazione ottimizzata per l'I/O che è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O che richiedono prezzi prevedibili con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

    L'archiviazione ottimizzata per l'I/O di Neptune è disponibile solo a partire dal rilascio 1.3.0.0 del motore.

- VpcSecurityGroups: una matrice di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco dei gruppi di sicurezza VPC a cui appartiene il cluster di database.

Errori

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- InvalidDBInstanceStateFault

AddRoleToDBCluster (operazione)

Il nome AWS CLI per questa API è: `add-role-to-db-cluster`.

Associa un ruolo Identity and Access Management (IAM) a un cluster database Neptune.

Richiesta

- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del cluster di database a cui associare il ruolo IAM.
- FeatureName (nella CLI: `--feature-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della funzionalità per il cluster database Neptune a cui associare il ruolo IAM. Per l'elenco dei nomi delle caratteristiche supportate, consulta the section called “DBEngineVersion”.
- RoleArn (nella CLI: `--role-arn`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) del ruolo IAM da associare al cluster di database Neptune, ad esempio `arn:aws:iam::123456789012:role/NeptuneAccessRole`.

Risposta

- Nessun parametro di risposta.

Errori

- DBClusterNotFoundFault
- DBClusterRoleAlreadyExistsFault
- InvalidDBClusterStateFault
- DBClusterRoleQuotaExceededFault

RemoveRoleFromDBCluster (operazione)

Il nome AWS CLI per questa API è: `remove-role-from-db-cluster`.

Annulla l'associazione di un ruolo Identity and Access Management (IAM) da un cluster di database.

Richiesta

- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

Nome del cluster di database di cui annullare l'associazione dal ruolo IAM.

- FeatureName (nella CLI: `--feature-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della funzionalità per il cluster database da cui si desidera dissociare il ruolo IAM. Per l'elenco dei nomi delle caratteristiche supportate, consulta the section called “DescribeDBEngineVersions”.
- RoleArn (nella CLI: `--role-arn`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) del ruolo IAM di cui annullare l'associazione dal cluster di database, ad esempio `arn:aws:iam::123456789012:role/NeptuneAccessRole`.

Risposta

- Nessun parametro di risposta.

Errori

- DBClusterNotFoundFault
- DBClusterRoleNotFoundFault
- InvalidDBClusterStateFault

FailoverDBCluster (operazione)

Il nome AWS CLI per questa API è: `failover-db-cluster`.

Forza un failover per un cluster di database.

Un failover di un cluster di database promuove una delle repliche di lettura (istanze di sola lettura) nel cluster di database a istanza primaria (cluster di scrittura).

Amazon Neptune esegue automaticamente il failover a una replica di lettura, se disponibile, in caso di errore dell'istanza primaria. Puoi forzare un failover per simulare un guasto di un'istanza primaria per scopi di testing. Poiché ogni istanza in un cluster di database ha un proprio indirizzo endpoint, devi eliminare e ristabilire tutte le connessioni esistenti che utilizzano tali indirizzi endpoint una volta completato il failover.

Richiesta

- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore del cluster di database per cui forzare un failover. Questo parametro non fa distinzione tra maiuscole e minuscole.

  Vincoli:

  - Deve corrispondere all'identificatore di un oggetto DBCluster esistente.
- TargetDBInstanceIdentifier (nella CLI: `--target-db-instance-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome dell'istanza da promuovere a istanza primaria.

  È necessario specificare l'identificatore di istanza per una replica di lettura nel cluster di database. Ad esempio, `mydbcluster-replica1`.

Risposta

Contiene i dettagli di un cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'the section called “DescribeDBClusters”.

- AllocatedStorage: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  `AllocatedStorage` restituisce sempre 1, perché le dimensioni di storage dei cluster di database Neptune non sono fisse, ma vengono regolate automaticamente in base alle esigenze.
- AssociatedRoles: una matrice di oggetti DBClusterRole.

  Fornisce un elenco dei ruoli Amazon Identity and Access Management (IAM) associati al cluster di database. I ruoli IAM associati a un cluster di database concedono l'autorizzazione per l'accesso del cluster di database ad altri servizi Amazon per tuo conto.
- AutomaticRestartTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Ora in cui il cluster database verrà riavviato automaticamente.
- AvailabilityZones: una stringa di tipo `string` (una stringa con codifica UTF-8).

Fornisce l'elenco delle zone di disponibilità EC2 in cui è possibile creare istanze nel cluster di database.

- BacktrackConsumedChangeRecords: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.

- BacktrackWindow: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.

- BackupRetentionPeriod: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.

- Capacity: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Non supportato da Neptune.

- CloneGroupId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identifica il gruppo di cloni a cui è associato il cluster di database.

- ClusterCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).

- CopyTagsToSnapshot: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su *`true`*, i tag vengono copiati in qualsiasi snapshot del cluster database che viene creato.

- CrossAccountClone: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `true`, il cluster database può essere clonato su più account.

- DatabaseName: una stringa di tipo `string` (una stringa con codifica UTF-8).

Contiene il nome del database iniziale di questo cluster di database, fornito al momento della creazione, se è stato specificato un nome quando il cluster di database è stato creato. Questo stesso nome viene restituito per la durata del cluster di database.

- DBClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per il cluster di database.

- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore del cluster di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un cluster di database.

- DBClusterMembers: una matrice di oggetti DBClusterMember.

  Fornisce l'elenco delle istanze che compongono il cluster di database.

- DBClusterParameterGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del gruppo di parametri del cluster di database per il cluster stesso.

- DbClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore univoco e non modificabile della regione Amazon per il cluster database. Questo identificativo è disponibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per il cluster database.

- DBSubnetGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica le informazioni sul gruppo di sottoreti associato al cluster di database, tra cui nome, descrizione e sottoreti nel gruppo.

- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se per il cluster di database è abilitata o meno la protezione da eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione.

- EarliestBacktrackTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Non supportato da Neptune.

- EarliestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica il primo orario possibile per il ripristino point-in-time di un database.

- EnabledCloudwatchLogsExports: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questo cluster database può esportare in CloudWatch Logs. I tipi di log validi sono: `audit` (per pubblicare log di audit su CloudWatch) e slowquery (per pubblicare log di slow query su CloudWatch). Consulta Pubblicazione di log Neptune nei file di log Amazon CloudWatch.

- Endpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'endpoint di connessione per l'istanza primaria del cluster di database.

- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questo cluster di database.

- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.

- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.

- HostedZoneId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'ID che Amazon Route 53 assegna al momento della creazione di una zona ospitata.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.

- IOOptimizedNextAllowedModificationTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La prossima volta puoi modificare il cluster di database per utilizzare il tipo di archiviazione `iopt1`.

- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per il cluster di database crittografato.

- LatestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.

- MultiAZ: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database ha istanze in più zone di disponibilità.

- PendingModifiedValues: un oggetto ClusterPendingModifiedValues.

  Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione `ModifyDBCluster` e contiene le modifiche che verranno applicate nella finestra di manutenzione successiva.

- PercentProgress: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'avanzamento dell'operazione sotto forma di percentuale.

- Port: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto il motore di database.

- PreferredBackupWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.

- ReaderEndpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'endpoint di lettura per il cluster di database. L'endpoint di lettura per un cluster di database bilancia il carico delle connessioni tra le repliche di lettura disponibili in un cluster di database. Man mano che i client richiedono nuove connessioni all'endpoint di lettura, Neptune distribuisce tali richieste fra le repliche di lettura nel cluster di database. Questa funzionalità è utile per bilanciare il carico di lavoro di lettura tra più repliche di lettura nel cluster di database.

  Se si verifica un failover e la replica di lettura a cui sei connesso viene promossa a istanza primaria, la connessione viene interrotta. Per continuare a inviare il carico di lavoro di lettura ad altre repliche di lettura nel cluster, puoi quindi riconnetterti all'endpoint di lettura.

- ReadReplicaIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

Contiene uno o più identificatori delle repliche di lettura associate a questo cluster di database.

- ReplicationSourceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ReplicationType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ServerlessV2ScalingConfiguration: un oggetto ServerlessV2ScalingConfigurationInfo.

  Mostra la configurazione del dimensionamento per un cluster database Neptune Serverless.

  Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del cluster di database.

- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database è crittografato.

- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione utilizzato dal cluster di database.

  Valori validi:

  - **`standard`**: (impostazione predefinita) fornisce archiviazione di database dai costi contenuti per applicazioni con un numero di operazioni di I/O da moderato a basso.
  - **`iopt1`**: abilita l'archiviazione ottimizzata per l'I/O che è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O che richiedono prezzi prevedibili con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

    L'archiviazione ottimizzata per l'I/O di Neptune è disponibile solo a partire dal rilascio 1.3.0.0 del motore.

- VpcSecurityGroups: una matrice di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco dei gruppi di sicurezza VPC a cui appartiene il cluster di database.

Errori

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- InvalidDBInstanceStateFault

PromoteReadReplicaDBCluster (operazione)

Il nome AWS CLI per questa API è: `promote-read-replica-db-cluster`.

Non supportato.

Richiesta

- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato.

Risposta

Contiene i dettagli di un cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'the section called “DescribeDBClusters”.

- AllocatedStorage: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  `AllocatedStorage` restituisce sempre 1, perché le dimensioni di storage dei cluster di database Neptune non sono fisse, ma vengono regolate automaticamente in base alle esigenze.
- AssociatedRoles: una matrice di oggetti DBClusterRole.

  Fornisce un elenco dei ruoli Amazon Identity and Access Management (IAM) associati al cluster di database. I ruoli IAM associati a un cluster di database concedono l'autorizzazione per l'accesso del cluster di database ad altri servizi Amazon per tuo conto.
- AutomaticRestartTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Ora in cui il cluster database verrà riavviato automaticamente.

- AvailabilityZones: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 in cui è possibile creare istanze nel cluster di database.

- BacktrackConsumedChangeRecords: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.

- BacktrackWindow: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.

- BackupRetentionPeriod: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.

- Capacity: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Non supportato da Neptune.

- CloneGroupId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identifica il gruppo di cloni a cui è associato il cluster di database.

- ClusterCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).

- CopyTagsToSnapshot: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su *`true`*, i tag vengono copiati in qualsiasi snapshot del cluster database che viene creato.

- CrossAccountClone: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `true`, il cluster database può essere clonato su più account.

- DatabaseName: una stringa di tipo `string` (una stringa con codifica UTF-8).

Contiene il nome del database iniziale di questo cluster di database, fornito al momento della creazione, se è stato specificato un nome quando il cluster di database è stato creato. Questo stesso nome viene restituito per la durata del cluster di database.

- DBClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per il cluster di database.

- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore del cluster di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un cluster di database.

- DBClusterMembers: una matrice di oggetti DBClusterMember.

  Fornisce l'elenco delle istanze che compongono il cluster di database.

- DBClusterParameterGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del gruppo di parametri del cluster di database per il cluster stesso.

- DbClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore univoco e non modificabile della regione Amazon per il cluster database. Questo identificativo è disponibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per il cluster database.

- DBSubnetGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica le informazioni sul gruppo di sottoreti associato al cluster di database, tra cui nome, descrizione e sottoreti nel gruppo.

- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se per il cluster di database è abilitata o meno la protezione da eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione.

- EarliestBacktrackTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Non supportato da Neptune.

- EarliestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica il primo orario possibile per il ripristino point-in-time di un database.

- EnabledCloudwatchLogsExports: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questo cluster database può esportare in CloudWatch Logs. I tipi di log validi sono: `audit` (per pubblicare log di audit su CloudWatch) e slowquery (per pubblicare log di slow query su CloudWatch). Consulta Pubblicazione di log Neptune nei file di log Amazon CloudWatch.

- Endpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'endpoint di connessione per l'istanza primaria del cluster di database.

- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questo cluster di database.

- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.

- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.

- HostedZoneId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'ID che Amazon Route 53 assegna al momento della creazione di una zona ospitata.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.

- IOOptimizedNextAllowedModificationTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La prossima volta puoi modificare il cluster di database per utilizzare il tipo di archiviazione `iopt1`.

- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per il cluster di database crittografato.

- LatestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.

- MultiAZ: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database ha istanze in più zone di disponibilità.

- PendingModifiedValues: un oggetto ClusterPendingModifiedValues.

  Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione `ModifyDBCluster` e contiene le modifiche che verranno applicate nella finestra di manutenzione successiva.

- PercentProgress: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'avanzamento dell'operazione sotto forma di percentuale.

- Port: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto il motore di database.

- PreferredBackupWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.

- ReaderEndpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'endpoint di lettura per il cluster di database. L'endpoint di lettura per un cluster di database bilancia il carico delle connessioni tra le repliche di lettura disponibili in un cluster di database. Man mano che i client richiedono nuove connessioni all'endpoint di lettura, Neptune distribuisce tali richieste fra le repliche di lettura nel cluster di database. Questa funzionalità è utile per bilanciare il carico di lavoro di lettura tra più repliche di lettura nel cluster di database.

  Se si verifica un failover e la replica di lettura a cui sei connesso viene promossa a istanza primaria, la connessione viene interrotta. Per continuare a inviare il carico di lavoro di lettura ad altre repliche di lettura nel cluster, puoi quindi riconnetterti all'endpoint di lettura.

- ReadReplicaIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

Contiene uno o più identificatori delle repliche di lettura associate a questo cluster di database.

- ReplicationSourceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ReplicationType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ServerlessV2ScalingConfiguration: un oggetto ServerlessV2ScalingConfigurationInfo.

  Mostra la configurazione del dimensionamento per un cluster database Neptune Serverless.

  Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del cluster di database.

- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database è crittografato.

- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione utilizzato dal cluster di database.

  Valori validi:

  - **`standard`**: (impostazione predefinita) fornisce archiviazione di database dai costi contenuti per applicazioni con un numero di operazioni di I/O da moderato a basso.
  - **`iopt1`**: abilita l'archiviazione ottimizzata per l'I/O che è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O che richiedono prezzi prevedibili con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

    L'archiviazione ottimizzata per l'I/O di Neptune è disponibile solo a partire dal rilascio 1.3.0.0 del motore.

- VpcSecurityGroups: una matrice di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco dei gruppi di sicurezza VPC a cui appartiene il cluster di database.

Errori

- DBClusterNotFoundFault
- InvalidDBClusterStateFault

DescribeDBClusters (operazione)

Il nome AWS CLI per questa API è: `describe-db-clusters`.

Restituisce informazioni sui cluster DB di provisioning e supporta l'impaginazione.

> **Note**
>
> Questa operazione può anche restituire informazioni per cluster Amazon RDS e cluster Amazon DocDB.

Richiesta

- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore del cluster di database fornito dall'utente. Se questo parametro viene specificato, vengono restituite solo le informazioni del cluster di database specifico. Questo parametro non fa distinzione tra maiuscole e minuscole.

  Vincoli:

  - Se viene specificato, deve corrispondere a un oggetto DBClusterIdentifier esistente.
- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Filtro che specifica uno o più cluster di database da descrivere.

  Filtri supportati:

  - `db-cluster-id`: accetta identificatori e Amazon Resource Name (ARN) di cluster di database. L'elenco di risultati includerà solo le informazioni sui cluster di database identificati da questi ARN.
  - `engine`: accetta un nome del motore (ad esempio `neptune`) e limita l'elenco dei risultati ai cluster database creati da tale motore.

Ad esempio, per richiamare questa API dall'interfaccia della riga di comando Amazon e filtrare in modo che vengano restituiti solo i cluster database Neptune, potresti utilizzare il seguente comando:

Example

```
aws neptune describe-db-clusters \
            --filters  Name=engine,Values=neptune
```

- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta the section called “DescribeDBClusters” precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.

Risposta

- DBClusters: una matrice di oggetti DBCluster.

  Contiene un elenco di cluster di database per l'utente.

- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione che è possibile utilizzare in una richiesta DescribeDBClusters successiva.

Errori

- DBClusterNotFoundFault

Strutture:

DBCluster (struttura)

Contiene i dettagli di un cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'the section called “DescribeDBClusters”.

Campi

- AllocatedStorage: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  `AllocatedStorage` restituisce sempre 1, perché le dimensioni di storage dei cluster di database Neptune non sono fisse, ma vengono regolate automaticamente in base alle esigenze.
- AssociatedRoles: questo è un array di oggetti DBClusterRole.

  Fornisce un elenco dei ruoli Amazon Identity and Access Management (IAM) associati al cluster di database. I ruoli IAM associati a un cluster di database concedono l'autorizzazione per l'accesso del cluster di database ad altri servizi Amazon per tuo conto.
- AutomaticRestartTime: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Ora in cui il cluster database verrà riavviato automaticamente.
- AvailabilityZones: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 in cui è possibile creare istanze nel cluster di database.
- BacktrackConsumedChangeRecords: questo è un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.
- BacktrackWindow: questo è un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.
- BackupRetentionPeriod: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.

- Capacity: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Non supportato da Neptune.

- CloneGroupId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identifica il gruppo di cloni a cui è associato il cluster di database.

- ClusterCreateTime: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).

- CopyTagsToSnapshot: questo è un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su *`true`*, i tag vengono copiati in qualsiasi snapshot del cluster database che viene creato.

- CrossAccountClone: questo è un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `true`, il cluster database può essere clonato su più account.

- DatabaseName: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nome del database iniziale di questo cluster di database, fornito al momento della creazione, se è stato specificato un nome quando il cluster di database è stato creato. Questo stesso nome viene restituito per la durata del cluster di database.

- DBClusterArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per il cluster di database.

- DBClusterIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore del cluster di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un cluster di database.

- DBClusterMembers: questo è un array di oggetti DBClusterMember.

  Fornisce l'elenco delle istanze che compongono il cluster di database.

- DBClusterParameterGroup: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

Specifica il nome del gruppo di parametri del cluster di database per il cluster stesso.

- DbClusterResourceId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore univoco e non modificabile della regione Amazon per il cluster database. Questo identificativo è disponibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per il cluster database.
- DBSubnetGroup: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica le informazioni sul gruppo di sottoreti associato al cluster di database, tra cui nome, descrizione e sottoreti nel gruppo.
- DeletionProtection: questo è un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se per il cluster di database è abilitata o meno la protezione da eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione.
- EarliestBacktrackTime: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Non supportato da Neptune.
- EarliestRestorableTime: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica il primo orario possibile per il ripristino point-in-time di un database.
- EnabledCloudwatchLogsExports: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questo cluster database può esportare in CloudWatch Logs. I tipi di log validi sono: `audit` (per pubblicare log di audit su CloudWatch) e slowquery (per pubblicare log di slow query su CloudWatch). Consulta Pubblicazione di log Neptune nei file di log Amazon CloudWatch.
- Endpoint: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'endpoint di connessione per l'istanza primaria del cluster di database.
- Engine: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questo cluster di database.
- EngineVersion: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

Indica la versione del motore di database.

- GlobalClusterIdentifier: questo è un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.
- HostedZoneId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'ID che Amazon Route 53 assegna al momento della creazione di una zona ospitata.
- IAMDatabaseAuthenticationEnabled: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.
- IOOptimizedNextAllowedModificationTime: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La prossima volta puoi modificare il cluster di database per utilizzare il tipo di archiviazione `iopt1`.
- KmsKeyId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per il cluster di database crittografato.
- LatestRestorableTime: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.
- MultiAZ: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database ha istanze in più zone di disponibilità.
- PendingModifiedValues: questo è un oggetto ClusterPendingModifiedValues.

  Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione `ModifyDBCluster` e contiene le modifiche che verranno applicate nella finestra di manutenzione successiva.
- PercentProgress: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'avanzamento dell'operazione sotto forma di percentuale.

- Port: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto il motore di database.

- PreferredBackupWindow: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.

- ReaderEndpoint: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'endpoint di lettura per il cluster di database. L'endpoint di lettura per un cluster di database bilancia il carico delle connessioni tra le repliche di lettura disponibili in un cluster di database. Man mano che i client richiedono nuove connessioni all'endpoint di lettura, Neptune distribuisce tali richieste fra le repliche di lettura nel cluster di database. Questa funzionalità è utile per bilanciare il carico di lavoro di lettura tra più repliche di lettura nel cluster di database.

  Se si verifica un failover e la replica di lettura a cui sei connesso viene promossa a istanza primaria, la connessione viene interrotta. Per continuare a inviare il carico di lavoro di lettura ad altre repliche di lettura nel cluster, puoi quindi riconnetterti all'endpoint di lettura.

- ReadReplicaIdentifiers: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori delle repliche di lettura associate a questo cluster di database.

- ReplicationSourceIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ReplicationType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ServerlessV2ScalingConfiguration: questo è un oggetto ServerlessV2ScalingConfigurationInfo.

  Mostra la configurazione del dimensionamento per un cluster database Neptune Serverless.

  Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

- Status: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del cluster di database.

- StorageEncrypted: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database è crittografato.

- StorageType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione utilizzato dal cluster di database.

  Valori validi:

  - **`standard`**: (impostazione predefinita) fornisce archiviazione di database dai costi contenuti per applicazioni con un numero di operazioni di I/O da moderato a basso.
  - **`iopt1`**: abilita l'archiviazione ottimizzata per l'I/O che è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O che richiedono prezzi prevedibili con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

    L'archiviazione ottimizzata per l'I/O di Neptune è disponibile solo a partire dal rilascio 1.3.0.0 del motore.

- VpcSecurityGroups: questo è un array di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco dei gruppi di sicurezza VPC a cui appartiene il cluster di database.

`DBCluster` viene utilizzato come elemento di risposta per:

- CreateDBCluster
- DeleteDBCluster
- FailoverDBCluster
- ModifyDBCluster
- PromoteReadReplicaDBCluster
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime
- StartDbCluster
- StopDBCluster

DBClusterMember (struttura)

Contiene informazioni su un'istanza che fa parte di un cluster di database.

Campi

- DBClusterParameterGroupStatus: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato del gruppo di parametri del cluster di database per il membro del cluster di database.
- DBInstanceIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'identificatore di istanza per il membro del cluster di database.
- IsClusterWriter: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Valore corrispondente a `true` se il membro del cluster è l'istanza primaria per il cluster di database, in caso contrario `false`.
- PromotionTier: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Valore che specifica l'ordine di promozione di una replica di lettura a istanza primaria dopo un errore dell'istanza primaria esistente.

DBClusterRole (struttura)

Descrive un ruolo Amazon Identity and Access Management (IAM) associato a un cluster di database.

Campi

- FeatureName: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della caratteristica associata al ruolo Amazon Identity and Access Management (IAM). Per l'elenco dei nomi delle caratteristiche supportate, consulta the section called “DescribeDBEngineVersions”.
- RoleArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) del ruolo IAM associato al cluster di database.

- Status: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Descrive lo stato dell'associazione tra il ruolo IAM e il cluster di database. La proprietà Status restituisce uno dei valori seguenti:

  - `ACTIVE`: l'ARN del ruolo IAM è associato al cluster database e può essere utilizzato per accedere ad altri servizi Amazon per tuo conto.
  - `PENDING`: è in corso l'associazione dell'ARN del ruolo IAM al cluster di database.
  - `INVALID`: l'ARN del ruolo IAM è associato al cluster database, ma il cluster database non è in grado di assumere il ruolo IAM per accedere ad altri servizi Amazon per tuo conto.

CloudwatchLogsExportConfiguration (struttura)

L'impostazione di configurazione per i tipi di log per consentire l'esportazione in CloudWatch Logs per una determinata istanza di database o un determinato cluster di database.

Le matrici `EnableLogTypes` e `DisableLogTypes` determinano quali log verranno esportati (o non esportati) in CloudWatch Logs.

I tipi di log validi sono: `audit` (per pubblicare log di audit) e `slowquery` (per pubblicare log di slow query). Consulta Pubblicazione di log Neptune nei file di log Amazon CloudWatch.

Campi

- DisableLogTypes: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'elenco dei tipi di log da disabilitare.
- EnableLogTypes: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'elenco dei tipi di log da abilitare.

PendingCloudwatchLogsExports (struttura)

Elenco dei tipi di log la cui configurazione è ancora in sospeso. In altre parole, questi tipi di log sono in fase di attivazione o disattivazione.

I tipi di log validi sono: `audit` (per pubblicare log di audit) e `slowquery` (per pubblicare log di slow query). Consulta Pubblicazione di log Neptune nei file di log Amazon CloudWatch.

Campi

- LogTypesToDisable: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Tipi di log in fase di abilitazione. Una volta abilitati, questi tipi di log vengono esportati in CloudWatch Logs.
- LogTypesToEnable: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Tipi di log in fase di disattivazione. Una volta disattivati, questi tipi di log non vengono esportati in CloudWatch Logs.

ClusterPendingModifiedValues (struttura)

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione `ModifyDBCluster` e contiene le modifiche che verranno applicate nella finestra di manutenzione successiva.

Campi

- AllocatedStorage: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  La dimensione dell'archiviazione allocato in gibibyte (GiB) per i motori di database. Per Neptune, `AllocatedStorage` restituisce sempre 1, perché le dimensioni di archiviazione dei cluster database Neptune non sono fisse, ma vengono regolate automaticamente in base alle esigenze.
- BackupRetentionPeriod: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Il numero di giorni durante i quali vengono conservati gli snapshot automatici del database.
- DBClusterIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il valore DBClusterIdentifier per il cluster database.
- EngineVersion: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  La versione del motore del database.
- IAMDatabaseAuthenticationEnabled: questo è un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica la mappatura degli account AWS Identity and Access Management (IAM) agli account di database è abilitata.

- Iops: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Il valore della capacità di IOPS (operazioni di I/O al secondo) allocata. Questa impostazione è solo per cluster di database Multi-AZ.

- PendingCloudwatchLogsExports: questo è un oggetto PendingCloudwatchLogsExports.

  Questa struttura `PendingCloudwatchLogsExports` specifica su quali log di CloudWatch sono abilitate e disabilitate le modifiche in sospeso.

- StorageType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  La modifica in sospeso nel tipo di archiviazione per il cluster di database. Valori validi:

  - **`standard`**: (impostazione predefinita) consente di configurare archiviazione di database dai costi contenuti per applicazioni con un numero di operazioni di I/O da moderato a basso.
  - **`iopt1`**: abilita l'archiviazione ottimizzata per l'I/O che è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O che richiedono prezzi prevedibili con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

    L'archiviazione ottimizzata per l'I/O di Neptune è disponibile solo a partire dal rilascio 1.3.0.0 del motore.

API del database globale Neptune

Operazioni:

- CreateGlobalCluster (azione)
- DeleteGlobalCluster (azione)
- ModifyGlobalCluster (azione)
- DescribeGlobalClusters (azione)
- FailoverGlobalCluster (azione)
- RemoveFromGlobalCluster (azione)

Strutture:

- GlobalCluster (struttura)
- GlobalClusterMember (struttura)

CreateGlobalCluster (azione)

Il nome AWS CLI per questa API è: `create-global-cluster`.

Crea un database globale Neptune distribuito in più regioni Amazon. Il database globale contiene un singolo cluster primario con funzionalità di lettura-scrittura e cluster secondari di sola lettura che ricevono i dati dal cluster primario tramite la replica ad alta velocità eseguita dal sottosistema di archiviazione Neptune.

È possibile creare un database globale inizialmente vuoto e poi aggiungervi un cluster primario e cluster secondari oppure specificare un cluster Neptune esistente durante l'operazione di creazione per trasformarlo nel cluster primario del database globale.

Richiesta

- DatabaseName (nella CLI: `--database-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome per il nuovo database globale (massimo di 64 caratteri alfanumerici).
- DeletionProtection (nella CLI: `--deletion-protection`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  L'impostazione di protezione contro l'eliminazione per il nuovo database globale. Il database globale non può essere eliminato quando è abilitata la protezione contro l'eliminazione.
- Engine (nella CLI: `--engine`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del motore di database da utilizzare nel database globale.

  Valori validi: `neptune`
- EngineVersion (nella CLI: `--engine-version`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  La versione del motore Neptune da utilizzare dal database globale.

  Valori validi: `1.2.0.0` o superiore.
- GlobalClusterIdentifier (nella CLI: `--global-cluster-identifier`): Obbligatorio: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Identificatore del cluster del nuovo cluster database globale.

- SourceDBClusterIdentifier (nella CLI: `--source-db-cluster-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  (Facoltativo) Il nome della risorsa Amazon (ARN) di un cluster database Neptune esistente da utilizzare come cluster primario del nuovo database globale.
- StorageEncrypted (nella CLI: `--storage-encrypted`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  L'impostazione della crittografia di archiviazione per il nuovo cluster database globale.

Risposta

Contiene i dettagli di un database globale Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta per le azioni the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” e the section called “RemoveFromGlobalCluster”.

- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  L'impostazione di protezione contro l'eliminazione per il database globale.
- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il motore di database Neptune utilizzato dal database globale (`"neptune"`).
- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La versione del motore Neptune utilizzata dal database globale.
- GlobalClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della risorsa Amazon (ARN) per il database globale.
- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.
- GlobalClusterMembers: una matrice di oggetti GlobalClusterMember.

Un elenco di ARN di cluster e ARN di istanze per tutti i cluster database che fanno parte del database globale.

- GlobalClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un identificatore immutabile per il database globale, univoco in tutte le regioni. Questo identificativo è disponibile nelle voci di log di CloudTrail ogni volta che si accede alla chiave KMS per il cluster database.
- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del database globale.
- StorageEncrypted: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  L'impostazione della crittografia di storage per il database globale.

Errori

- GlobalClusterAlreadyExistsFault
- GlobalClusterQuotaExceededFault
- InvalidDBClusterStateFault
- DBClusterNotFoundFault

DeleteGlobalCluster (azione)

Il nome AWS CLI per questa API è: `delete-global-cluster`.

Elimina un database globale. Il cluster primario e tutti i cluster secondari devono essere già stati scollegati o eliminati.

Richiesta

- GlobalClusterIdentifier (nella CLI: `--global-cluster-identifier`): Obbligatorio: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  L'identificatore del cluster del nuovo cluster database globale da eliminare.

Risposta

Contiene i dettagli di un database globale Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta per le azioni the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” e the section called “RemoveFromGlobalCluster”.

- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  L'impostazione di protezione contro l'eliminazione per il database globale.
- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il motore di database Neptune utilizzato dal database globale (`"neptune"`).
- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La versione del motore Neptune utilizzata dal database globale.
- GlobalClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della risorsa Amazon (ARN) per il database globale.
- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.
- GlobalClusterMembers: una matrice di oggetti GlobalClusterMember.

  Un elenco di ARN di cluster e ARN di istanze per tutti i cluster database che fanno parte del database globale.
- GlobalClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un identificatore immutabile per il database globale, univoco in tutte le regioni. Questo identificativo è disponibile nelle voci di log di CloudTrail ogni volta che si accede alla chiave KMS per il cluster database.
- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del database globale.
- StorageEncrypted: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

L'impostazione della crittografia di storage per il database globale.

Errori

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

ModifyGlobalCluster (azione)

Il nome AWS CLI per questa API è: `modify-global-cluster`.

Modifica un'impostazione per un cluster globale di Amazon Neptune. Puoi modificare uno o più parametri di configurazione del database specificando questi parametri e i nuovi valori nella richiesta.

Richiesta

- AllowMajorVersionUpgrade (nella CLI: `--allow-major-version-upgrade`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che Indica che gli aggiornamenti delle versioni principali sono permessi.

  Vincoli: gli aggiornamenti delle versioni principali devono essere consentiti se specifichi un valore per il parametro `EngineVersion` con una versione principale diversa rispetto alla versione corrente dell'istanza database.

  Se aggiorni la versione principale di un database globale, i gruppi di parametri del cluster e dell'istanza database vengono impostati sui gruppi di parametri predefiniti per la nuova versione, quindi sarà necessario applicare eventuali gruppi di parametri personalizzati dopo aver completato l'aggiornamento.
- DeletionProtection (nella CLI: `--deletion-protection`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se per il database globale è abilitata o meno la protezione da eliminazione. Il database globale non può essere eliminato quando è abilitata la protezione da eliminazione.
- EngineVersion (nella CLI: `--engine-version`): una stringa di tipo `string` (una stringa con codifica UTF-8).

Numero di versione del motore di database a cui eseguire l'aggiornamento. La modifica di questo parametro provoca un'interruzione. La modifica viene applicata durante la finestra di manutenzione successiva, a meno che il parametro `ApplyImmediately` non sia abilitato.

Per ottenere un elenco di tutte le versioni dei motori Neptune disponibili, utilizza il comando seguente:

Example

```
aws neptune describe-db-engine-versions \
          --engine neptune \
          --query '*[]|[?SupportsGlobalDatabases == 'true'].[EngineVersion]'
```

- GlobalClusterIdentifier (nella CLI: `--global-cluster-identifier`): Obbligatorio: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Identificatore del cluster database globale da modificare. Questo parametro non opera distinzione tra maiuscole e minuscole.

  Vincoli: deve corrispondere all'identificatore di un cluster database globale esistente.
- NewGlobalClusterIdentifier (nella CLI: `--new-global-cluster-identifier`): un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Un nuovo identificatore di cluster da assegnare al database globale. Questo valore è archiviato come stringa in caratteri minuscoli.

  Vincoli:

  - Deve contenere da 1 a 63 lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi

  Esempio: `my-cluster2`

Risposta

Contiene i dettagli di un database globale Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta per le azioni the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” e the section called “RemoveFromGlobalCluster”.

- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  L'impostazione di protezione contro l'eliminazione per il database globale.
- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il motore di database Neptune utilizzato dal database globale (`"neptune"`).
- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La versione del motore Neptune utilizzata dal database globale.
- GlobalClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della risorsa Amazon (ARN) per il database globale.
- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.
- GlobalClusterMembers: una matrice di oggetti GlobalClusterMember.

  Un elenco di ARN di cluster e ARN di istanze per tutti i cluster database che fanno parte del database globale.
- GlobalClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un identificatore immutabile per il database globale, univoco in tutte le regioni. Questo identificativo è disponibile nelle voci di log di CloudTrail ogni volta che si accede alla chiave KMS per il cluster database.
- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del database globale.
- StorageEncrypted: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  L'impostazione della crittografia di storage per il database globale.

Errori

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

DescribeGlobalClusters (azione)

Il nome AWS CLI per questa API è: `describe-global-clusters`.

Restituisce informazioni sui cluster database globali di Neptune. Quest'API supporta la paginazione.

Richiesta

- GlobalClusterIdentifier (nella CLI: `--global-cluster-identifier`): un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Identificatore del cluster database fornito dall'utente. Se questo parametro viene specificato, vengono restituite solo le informazioni del cluster database specificato. Questo parametro non opera distinzione tra maiuscole e minuscole.

  Vincoli: se viene specificato, deve corrispondere all'identificatore di un cluster database esistente.
- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  (Facoltativo) Token di paginazione restituito da una chiamata precedente a `DescribeGlobalClusters`. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al numero specificato da `MaxRecords`.
- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se esistono più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione che permette di recuperare i risultati rimanenti.

  Impostazione predefinita: `100`

  Vincoli: minimo 20, massimo 100.

Risposta

- GlobalClusters: una matrice di oggetti GlobalCluster.

  L'elenco dei cluster e delle istanze globali restituiti da questa richiesta.
- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un token di paginazione. Se questo parametro viene restituito nella risposta, sono disponibili più record, che possono essere recuperati con una o più chiamate aggiuntive a `DescribeGlobalClusters`.

Errori

- GlobalClusterNotFoundFault

FailoverGlobalCluster (azione)

Il nome AWS CLI per questa API è: `failover-global-cluster`.

Avvia il processo di failover per un database globale Neptune.

Un failover per un database globale Neptune promuove uno dei cluster database secondari di sola lettura in modo che diventi il cluster database primario. Inoltre, retrocede il cluster database primario a cluster database secondario (di sola lettura). In altre parole, il ruolo del cluster database primario corrente e del cluster database secondario di destinazione selezionato vengono scambiati. Il cluster database secondario selezionato assume funzionalità complete di lettura/scrittura per il database globale Neptune.

> **Note**
>
> Questa azione si applica solo ai database globali Neptune. Questa azione è destinata esclusivamente all'uso su database globali Neptune integri con cluster database Neptune integri e senza interruzioni a livello regionale, per testare scenari di ripristino di emergenza o per riconfigurare la topologia del database globale.

Richiesta

- GlobalClusterIdentifier (nella CLI: `--global-cluster-identifier`): Obbligatorio: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

Identificatore del database globale Neptune su cui eseguire il failover. L'identificatore è la chiave univoca assegnata dall'utente al momento della creazione del database globale Neptune. In altre parole, è il nome del database globale di cui eseguire il failover.

Vincoli: deve corrispondere all'identificatore di un cluster database globale esistente.

- TargetDbClusterIdentifier (nella CLI: `--target-db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della risorsa Amazon (ARN) del cluster database Neptune da promuovere in modo che diventi quello primario per il database globale.

Risposta

Contiene i dettagli di un database globale Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta per le azioni the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” e the section called “RemoveFromGlobalCluster”.

- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  L'impostazione di protezione contro l'eliminazione per il database globale.
- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il motore di database Neptune utilizzato dal database globale (`"neptune"`).
- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La versione del motore Neptune utilizzata dal database globale.
- GlobalClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della risorsa Amazon (ARN) per il database globale.
- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.

- GlobalClusterMembers: una matrice di oggetti GlobalClusterMember.

  Un elenco di ARN di cluster e ARN di istanze per tutti i cluster database che fanno parte del database globale.
- GlobalClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un identificatore immutabile per il database globale, univoco in tutte le regioni. Questo identificativo è disponibile nelle voci di log di CloudTrail ogni volta che si accede alla chiave KMS per il cluster database.
- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del database globale.
- StorageEncrypted: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  L'impostazione della crittografia di storage per il database globale.

Errori

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault
- InvalidDBClusterStateFault
- DBClusterNotFoundFault

RemoveFromGlobalCluster (azione)

> Il nome AWS CLI per questa API è: `remove-from-global-cluster`.

Scollega un cluster database Neptune da un database globale Neptune. Un cluster secondario diventa un normale cluster autonomo con funzionalità di lettura/scrittura, anziché essere di sola lettura. Inoltre, non riceve più dati dal cluster primario.

Richiesta

- DbClusterIdentifier (nella CLI: `--db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della risorsa Amazon (ARN) che identifica il cluster da scollegare dal cluster database globale Neptune.

- GlobalClusterIdentifier (nella CLI: `--global-cluster-identifier`): Obbligatorio: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  L'identificatore del database globale Neptune da cui scollegare il cluster database Neptune specificato.

Risposta

Contiene i dettagli di un database globale Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta per le azioni the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” e the section called “RemoveFromGlobalCluster”.

- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  L'impostazione di protezione contro l'eliminazione per il database globale.
- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il motore di database Neptune utilizzato dal database globale (`"neptune"`).
- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La versione del motore Neptune utilizzata dal database globale.
- GlobalClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della risorsa Amazon (ARN) per il database globale.
- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.
- GlobalClusterMembers: una matrice di oggetti GlobalClusterMember.

  Un elenco di ARN di cluster e ARN di istanze per tutti i cluster database che fanno parte del database globale.

- GlobalClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un identificatore immutabile per il database globale, univoco in tutte le regioni. Questo identificativo è disponibile nelle voci di log di CloudTrail ogni volta che si accede alla chiave KMS per il cluster database.
- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del database globale.
- StorageEncrypted: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  L'impostazione della crittografia di storage per il database globale.

Errori

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault
- DBClusterNotFoundFault

Strutture:

GlobalCluster (struttura)

Contiene i dettagli di un database globale Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta per le azioni the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” e the section called “RemoveFromGlobalCluster”.

Campi

- DeletionProtection: questo è un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  L'impostazione di protezione contro l'eliminazione per il database globale.
- Engine: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il motore di database Neptune utilizzato dal database globale (`"neptune"`).

- EngineVersion: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  La versione del motore Neptune utilizzata dal database globale.

- GlobalClusterArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della risorsa Amazon (ARN) per il database globale.

- GlobalClusterIdentifier: questo è un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.

- GlobalClusterMembers: questo è un array di oggetti GlobalClusterMember.

  Un elenco di ARN di cluster e ARN di istanze per tutti i cluster database che fanno parte del database globale.

- GlobalClusterResourceId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un identificatore immutabile per il database globale, univoco in tutte le regioni. Questo identificativo è disponibile nelle voci di log di CloudTrail ogni volta che si accede alla chiave KMS per il cluster database.

- Status: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del database globale.

- StorageEncrypted: questo è un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  L'impostazione della crittografia di storage per il database globale.

`GlobalCluster` viene utilizzato come elemento di risposta per:

- CreateGlobalCluster
- ModifyGlobalCluster
- DeleteGlobalCluster
- RemoveFromGlobalCluster
- FailoverGlobalCluster

GlobalClusterMember (struttura)

Una struttura di dati con informazioni su tutti i cluster primari e secondari associati a un database globale Neptune.

Campi

- DBClusterArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della risorsa Amazon (ARN) per ogni cluster Neptune.
- IsWriter: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster Neptune è quello primario (ovvero dispone di funzionalità di lettura/scrittura) per il database globale Neptune a cui è associato.
- Readers: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della risorsa Amazon (ARN) per ogni cluster secondario di sola lettura associato al database globale Neptune.

API per le istanze Neptune

Operazioni:

- CreateDBInstance (operazione)
- DeleteDBInstance (operazione)
- ModifyDBInstance (operazione)
- RebootDBInstance (operazione)
- DescribeDBInstances (operazione)
- DescribeOrderableDBInstanceOptions (operazione)
- DescribeValidDBInstanceModifications (operazione)

Strutture:

- DBInstance (struttura)
- DBInstanceStatusInfo (struttura)
- OrderableDBInstanceOption (struttura)

- PendingModifiedValues (struttura)
- ValidStorageOptions (struttura)
- ValidDBInstanceModificationsMessage (struttura)

CreateDBInstance (operazione)

Il nome AWS CLI per questa API è: `create-db-instance`.

Crea una nuova istanza database.

Richiesta

- AutoMinorVersionUpgrade (nella CLI: `--auto-minor-version-upgrade`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Indica che gli aggiornamenti secondari del motore sono applicati automaticamente all'istanza database durante la finestra di manutenzione.

  Impostazione predefinita: `true`
- AvailabilityZone (nella CLI: `--availability-zone`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Zona di disponibilità EC2 in cui viene creata l'istanza database

  Impostazione predefinita: una zona di disponibilità casuale scelta dal sistema nella regione Amazon dell'endpoint.

  Esempio: `us-east-1d`

  Vincolo: il parametro AvailabilityZone non può essere specificato se il parametro MultiAZ è impostato su `true`. La zona di disponibilità specificata deve trovarsi nella stessa regione Amazon dell'endpoint corrente.
- BackupRetentionPeriod (nella CLI: `--backup-retention-period`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Il numero di giorni durante i quali vengono conservati i backup automatici.

  Non applicabile. Il periodo di retention per i backup automatici viene gestito dal cluster di database. Per ulteriori informazioni, consulta the section called “CreateDBCluster”.

Impostazione predefinita: 1

Vincoli:

- Il valore deve essere compreso tra 0 e 35
- Il valore non può essere impostato su 0 se l'istanza database è un'origine di repliche di lettura

- CopyTagsToSnapshot (nella CLI: `--copy-tags-to-snapshot`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  True per copiare tutti i tag dall'istanza database agli snapshot dell'istanza database, in caso contrario false. Il valore di default è false.
- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore del cluster di database a cui apparterrà l'istanza.

  Per ulteriori informazioni sulla creazione di un cluster database, consulta the section called “CreateDBCluster”.

  Tipo: string
- DBInstanceClass (nella CLI: `--db-instance-class`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Capacità di calcolo e di memoria dell'istanza database, ad esempio `db.m4.large`. Non tutte le classi di istanza database sono disponibili in tutte le regioni Amazon.
- DBInstanceIdentifier (nella CLI: `--db-instance-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore istanze DB. Questo parametro è archiviato come stringa in minuscolo.

  Vincoli:

  - Deve contenere da 1 a 63 lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.

  Esempio: `mydbinstance`
- DBName (nella CLI: `--db-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

Non supportato.

- DBParameterGroupName (nella CLI: `--db-parameter-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del gruppo di parametri database da associare all'istanza database. Se questo argomento viene omesso, viene utilizzato l'oggetto DBParameterGroup predefinito per il motore specificato.

  Vincoli:

  - Deve contenere da 1 a 255 lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera
  - Non può terminare con un trattino o contenere due trattini consecutivi
- DBSecurityGroups (nella CLI: `--db-security-groups`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco dei gruppi di sicurezza DB da associare all'istanza database.

  Impostazione predefinita: il gruppo di sicurezza DB predefinito per il motore di database.
- DBSubnetGroupName (nella CLI: `--db-subnet-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Gruppo di sottoreti database da associare all'istanza database.

  Se non c'è un gruppo di sottoreti database, l'istanza non è un'istanza database VPC.
- DeletionProtection (nella CLI: `--deletion-protection`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se per l'istanza database è abilitata la protezione da eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione. Per impostazione predefinita, la protezione da eliminazione è disabilitata. Consultare Eliminazione di un'istanza database.

  Le istanze database in un cluster database possono essere eliminate anche quando per il cluster database è abilitata la protezione da eliminazione.
- Domain (nella CLI: `--domain`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il dominio di Active Directory in cui creare l'istanza.

- DomainIAMRoleName (nella CLI: `--domain-iam-role-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del ruolo IAM da utilizzare per le chiamate API al servizio directory.

- EnableCloudwatchLogsExports (nella CLI: `--enable-cloudwatch-logs-exports`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco dei tipi di log che devono essere abilitati per l'esportazione in CloudWatch Logs.

- EnableIAMDatabaseAuthentication (nella CLI: `--enable-iam-database-authentication`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Non supportato da Neptune (ignorato).

- Engine (nella CLI: `--engine`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del motore di database da utilizzare per questa istanza.

  Valori validi: `neptune`

- EngineVersion (nella CLI: `--engine-version`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Numero di versione del motore di database da utilizzare. Attualmente, l'impostazione di questo parametro non ha alcun effetto.

- Iops (nella CLI: `--iops`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Quantità di Provisioned IOPS (operazioni di input/output al secondo) da allocare inizialmente all'istanza database.

- KmsKeyId (nella CLI: `--kms-key-id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore della chiave Amazon KMS per un'istanza database crittografata.

  L'identificatore della chiave KMS è l'Amazon Resource Name (ARN) per la chiave di crittografia KMS. Se stai creando un'istanza database con lo stesso account Amazon che possiede la chiave di crittografia KMS utilizzata per crittografare la nuova istanza database, puoi utilizzare l'alias della chiave KMS al posto dell'ARN per la chiave di crittografia KM.

Non applicabile. L'identificatore della chiave KMS viene gestito dal cluster di database. Per ulteriori informazioni, consulta the section called “CreateDBCluster”.

Se il parametro `StorageEncrypted` è true e non specifichi un valore per il parametro `KmsKeyId`, Amazon Neptune utilizzerà la chiave di crittografia predefinita. Amazon KMS crea la chiave di crittografia predefinita per l'account Amazon. L'account Amazon ha una chiave crittografica predefinita diversa per ogni regione Amazon.

- LicenseModel (nella CLI: `--license-model`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Informazioni sul modello di licenza per questa istanza database.

  Valori validi: `license-included` | `bring-your-own-license` | `general-public-license`

- MonitoringInterval (nella CLI: `--monitoring-interval`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Intervallo, in secondi, tra i punti in cui vengono raccolti i parametri di monitoraggio avanzato per l'istanza database. Per disabilitare la raccolta dei parametri di monitoraggio avanzato, specifica 0. Il valore predefinito è 0.

  Se specifichi `MonitoringRoleArn`, devi anche impostare `MonitoringInterval` su un valore diverso da 0.

  Valori validi: `0, 1, 5, 10, 15, 30, 60`

- MonitoringRoleArn (nella CLI: `--monitoring-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN del ruolo IAM che permette a Neptune di inviare i parametri di monitoraggio avanzato ad Amazon CloudWatch Logs. Ad esempio, `arn:aws:iam:123456789012:role/emaccess`.

  Se `MonitoringInterval` è impostato su un valore diverso da 0, devi fornire un valore di `MonitoringRoleArn`.

- MultiAZ (nella CLI: `--multi-az`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se l'istanza database è un'implementazione Multi-AZ. Non è possibile impostare il parametro AvailabilityZone se il parametro MultiAZ è impostato su true.

- Port (nella CLI: `--port`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Il numero della porta sulla quale il database accetta connessioni.

  Non applicabile. La porta viene gestita dal cluster di database. Per ulteriori informazioni, consulta the section called “CreateDBCluster”.

  Default: `8182`

  Tipo: integer

- PreferredBackupWindow (nella CLI: `--preferred-backup-window`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Intervallo di tempo giornaliero durante il quale vengono creati i backup automatici.

  Non applicabile. L'intervallo di tempo giornaliero per la creazione dei backup automatici viene gestito dal cluster di database. Per ulteriori informazioni, consulta the section called “CreateDBCluster”.

- PreferredMaintenanceWindow (nella CLI: `--preferred-maintenance-window`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Intervallo di tempo settimanale durante il quale può venire eseguita la manutenzione del sistema, nel fuso orario UTC (Universal Coordinated Time).

  Formato: `ddd:hh24:mi-ddd:hh24:mi`

  Il valore predefinito è una finestra di 30 minuti selezionata in modo casuale in un blocco di 8 ore per ogni regione Amazon, in un giorno casuale della settimana.

  Giorni validi: lunedì, martedì, mercoledì, giovedì, venerdì, sabato, domenica.

  Vincoli: finestra di un minimo di 30 minuti.

- PromotionTier (nella CLI: `--promotion-tier`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Valore che specifica l'ordine di promozione di una replica di lettura a istanza primaria dopo un errore dell'istanza primaria esistente.

  Impostazione predefinita: 1

Valori validi: 0-15

- PubliclyAccessible (nella CLI: `--publicly-accessible`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Questo flag non deve più essere utilizzato.

- StorageEncrypted (nella CLI: `--storage-encrypted`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se l'istanza database è crittografata.

  Non applicabile. La crittografia per le istanze database viene gestita dal cluster di database. Per ulteriori informazioni, consulta the section called “CreateDBCluster”.

  Impostazione predefinita: false

- StorageType (nella CLI: `--storage-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non applicabile. In Neptune il tipo di archiviazione è gestito a livello di cluster di database.

- Tags (nella CLI: `--tags`): un array di oggetti Tag.

  Tag da assegnare alla nuova istanza.

- TdeCredentialArn (nella CLI: `--tde-credential-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN dell'archivio chiavi con cui associare l'istanza per la crittografia TDE.

- TdeCredentialPassword (nella CLI: `--tde-credential-password`): una SensitiveString di tipo `string` (una stringa con codifica UTF-8).

  Password per l'ARN specificato dall'archivio chiavi per accedere al dispositivo.

- Timezone (nella CLI: `--timezone`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fuso orario dell'istanza database.

- VpcSecurityGroupIds (nella CLI: `--vpc-security-group-ids`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco dei gruppi di sicurezza VPC EC2 da associare all'istanza database.

Non applicabile. L'elenco associato di gruppi di sicurezza VPC EC2 viene gestito dal cluster di database. Per ulteriori informazioni, consulta the section called “CreateDBCluster”.

Impostazione predefinita: il gruppo di sicurezza VPC EC2 predefinito per il VPC del gruppo di sottoreti database.

Risposta

Contiene i dettagli di un'istanza database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called “DescribeDBInstances”.

- AutoMinorVersionUpgrade: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica che le patch della versione secondaria vengono applicate automaticamente.
- AvailabilityZone: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome della zona di disponibilità in cui si trova l'istanza database.
- BackupRetentionPeriod: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.
- CACertificateIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore del certificato CA per questa istanza database.
- CopyTagsToSnapshot: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se i tag vengono copiati dall'istanza database agli snapshot dell'istanza database.
- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se l'istanza database è membro di un cluster di database, contiene il nome del cluster di database di cui l'istanza database è membro.
- DBInstanceArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per l'istanza database.
- DBInstanceClass: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nome della classe di capacità di calcolo e di memoria dell'istanza database.

- DBInstanceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un'istanza database.
- DbInstancePort: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto l'istanza database. Se l'istanza database fa parte di un cluster di database, questa porta può essere diversa rispetto a quella del cluster di database.
- DBInstanceStatus: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente di questo database.
- DbiResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore univoco e non modificabile della regione Amazon per l'istanza database. Questo identificatore è reperibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per l'istanza database.
- DBName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del database.
- DBParameterGroups: una matrice di oggetti DBParameterGroupStatus.

  Fornisce l'elenco dei gruppi di parametri database applicati a questa istanza database.
- DBSecurityGroups: una matrice di oggetti DBSecurityGroupMembership.

  Fornisce l'elenco degli elementi del gruppo di sicurezza DB contenenti solo elementi secondari `DBSecurityGroup.Name` e `DBSecurityGroup.Status`.
- DBSubnetGroup: un oggetto DBSubnetGroup.

  Specifica le informazioni sul gruppo di sottoreti associato all'istanza database, inclusi il nome, la descrizione e le sottoreti presenti nel gruppo.
- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se per l'istanza database è abilitata o meno la protezione da eliminazione. L'istanza non può essere eliminata se è abilitata la protezione da eliminazione. Consultare Eliminazione di un'istanza database.
- DomainMemberships: una matrice di oggetti DomainMembership.

  Non supportato

- EnabledCloudwatchLogsExports: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questa istanza database è configurata per esportare in CloudWatch Logs.
- Endpoint: un oggetto Endpoint.

  Specifica l'endpoint di connessione.
- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questa istanza database.
- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.
- EnhancedMonitoringResourceArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) del flusso di log Amazon CloudWatch Logs che riceve i dati dei parametri di monitoraggio avanzato per l'istanza database.
- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se l'autenticazione di Amazon Identity and Access Management (IAM) è abilitata, in caso contrario false.
- InstanceCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica data e ora di creazione dell'istanza database.
- Iops: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il valore di Provisioned IOPS (operazioni di I/O al secondo).
- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato: la crittografia per le istanze database viene gestita dal cluster di database.
- LatestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.
- LicenseModel: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Informazioni sul modello di licenza per questa istanza database.

- MonitoringInterval: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Intervallo, in secondi, tra i punti in cui vengono raccolti i parametri di monitoraggio avanzato per l'istanza database.
- MonitoringRoleArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN del ruolo IAM che permette a Neptune di inviare i parametri di monitoraggio avanzato ad Amazon CloudWatch Logs.
- MultiAZ: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se l'istanza database è un'implementazione Multi-AZ.
- PendingModifiedValues: un oggetto PendingModifiedValues.

  Specifica che le modifiche per l'istanza database sono in sospeso. Questo elemento è incluso solo quando sono presenti modifiche in sospeso. Le modifiche specifiche sono identificate dagli elementi secondari.
- PreferredBackupWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.
- PromotionTier: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Valore che specifica l'ordine di promozione di una replica di lettura a istanza primaria dopo un errore dell'istanza primaria esistente.
- PubliclyAccessible: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Questo flag non deve più essere utilizzato.
- ReadReplicaDBClusterIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori dei cluster di database che sono repliche di lettura di questa istanza database.
- ReadReplicaDBInstanceIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori delle repliche di lettura associate a questa istanza database.

- ReadReplicaSourceDBInstanceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene l'identificatore dell'istanza database di origine se questa istanza database è una replica di lettura.
- SecondaryAvailabilityZone: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se presente, specifica il nome della zona di disponibilità secondaria per un'istanza database con supporto per AZ multiple.
- StatusInfos: una matrice di oggetti DBInstanceStatusInfo.

  Stato di una replica di lettura. Se l'istanza non è una replica di lettura, questo campo è vuoto.
- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Non supportato: la crittografia per le istanze database viene gestita dal cluster di database.
- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il tipo di archiviazione associato all'istanza database.
- TdeCredentialArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN dell'archivio chiavi con cui l'istanza è associata per la crittografia TDE.
- Timezone: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato.
- VpcSecurityGroups: una matrice di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco di elementi del gruppo di sicurezza VPC a cui appartiene l'istanza database.

Errori

- DBInstanceAlreadyExistsFault
- InsufficientDBInstanceCapacityFault
- DBParameterGroupNotFoundFault
- DBSecurityGroupNotFoundFault
- InstanceQuotaExceededFault
- StorageQuotaExceededFault

- DBSubnetGroupNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidDBClusterStateFault
- InvalidSubnet
- InvalidVPCNetworkStateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBClusterNotFoundFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- KMSKeyNotAccessibleFault
- DomainNotFoundFault

DeleteDBInstance (operazione)

Il nome AWS CLI per questa API è: `delete-db-instance`.

L'operazione DeleteDBInstance elimina un'istanza database di cui è stato effettuato il provisioning in precedenza. Quando elimini un'istanza database, tutti i backup automatici per tale istanza vengono eliminati e non possono essere recuperati. Gli snapshot DB manuali dell'istanza database da eliminare tramite `DeleteDBInstance` non vengono eliminati.

Se richiedi uno snapshot DB finale, lo stato dell'istanza database Amazon Neptune è `deleting` fino a quando non viene completata la creazione dello snapshot DB. L'operazione API `DescribeDBInstance` viene utilizzata per monitorare lo stato di questa operazione. Una volta inviata, l'operazione non può essere annullata.

Quando un'istanza database si trova in stato di errore con lo stato `failed`, `incompatible-restore` o `incompatible-network`, è possibile eliminarla solo quando il parametro `SkipFinalSnapshot` è impostato su `true`.

Non è possibile eliminare un'istanza database se è l'unica istanza nel cluster database o se è attivata la protezione da eliminazione.

Richiesta

- DBInstanceIdentifier (nella CLI: `--db-instance-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore dell'istanza database da eliminare. Questo parametro non fa distinzione tra maiuscole e minuscole.

  Vincoli:

  - Deve corrispondere al nome di un'istanza database esistente.
- FinalDBSnapshotIdentifier (nella CLI: `--final-db-snapshot-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Oggetto DBSnapshotIdentifier del nuovo snapshot DB creato quando SkipFinalSnapshot è impostato su `false`.

  > **Note**
  >
  > Se specifichi questo parametro e imposti anche il parametro SkipFinalShapshot su true, viene generato un errore.

  Vincoli:

  - Deve contenere da 1 a 255 lettere o numeri.
  - Il primo carattere deve essere una lettera
  - Non può terminare con un trattino o contenere due trattini consecutivi
  - Non può essere specificato quando è in corso l'eliminazione di una replica di lettura.
- SkipFinalSnapshot (nella CLI: `--skip-final-snapshot`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Determina se viene creato uno snapshot DB finale prima dell'eliminazione dell'istanza database. Se viene specificato `true`, non viene creato alcuno snapshot DB. Se viene specificato `false`, viene creato uno snapshot DB prima dell'eliminazione dell'istanza database.

  Quando un'istanza database si trova in stato di errore con lo stato "failed", "incompatible-restore" o "incompatible-network", è possibile eliminarla solo quando il parametro SkipFinalSnapshot è impostato su "true".

  Specifica `true` quando è in corso l'eliminazione di una replica di lettura.

> **Note**
>
> È necessario specificare il parametro FinalDBSnapshotIdentifier se SkipFinalSnapshot è `false`.

Impostazione predefinita: `false`

Risposta

Contiene i dettagli di un'istanza database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called "DescribeDBInstances".

- AutoMinorVersionUpgrade: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica che le patch della versione secondaria vengono applicate automaticamente.
- AvailabilityZone: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome della zona di disponibilità in cui si trova l'istanza database.
- BackupRetentionPeriod: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.
- CACertificateIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore del certificato CA per questa istanza database.
- CopyTagsToSnapshot: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se i tag vengono copiati dall'istanza database agli snapshot dell'istanza database.
- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se l'istanza database è membro di un cluster di database, contiene il nome del cluster di database di cui l'istanza database è membro.
- DBInstanceArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per l'istanza database.

- DBInstanceClass: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nome della classe di capacità di calcolo e di memoria dell'istanza database.
- DBInstanceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un'istanza database.
- DbInstancePort: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto l'istanza database. Se l'istanza database fa parte di un cluster di database, questa porta può essere diversa rispetto a quella del cluster di database.
- DBInstanceStatus: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente di questo database.
- DbiResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore univoco e non modificabile della regione Amazon per l'istanza database. Questo identificatore è reperibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per l'istanza database.
- DBName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del database.
- DBParameterGroups: una matrice di oggetti DBParameterGroupStatus.

  Fornisce l'elenco dei gruppi di parametri database applicati a questa istanza database.
- DBSecurityGroups: una matrice di oggetti DBSecurityGroupMembership.

  Fornisce l'elenco degli elementi del gruppo di sicurezza DB contenenti solo elementi secondari `DBSecurityGroup.Name` e `DBSecurityGroup.Status`.
- DBSubnetGroup: un oggetto DBSubnetGroup.

  Specifica le informazioni sul gruppo di sottoreti associato all'istanza database, inclusi il nome, la descrizione e le sottoreti presenti nel gruppo.
- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se per l'istanza database è abilitata o meno la protezione da eliminazione. L'istanza non può essere eliminata se è abilitata la protezione da eliminazione. Consultare Eliminazione di un'istanza database.

- DomainMemberships: una matrice di oggetti DomainMembership.

  Non supportato

- EnabledCloudwatchLogsExports: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questa istanza database è configurata per esportare in CloudWatch Logs.

- Endpoint: un oggetto Endpoint.

  Specifica l'endpoint di connessione.

- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questa istanza database.

- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.

- EnhancedMonitoringResourceArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) del flusso di log Amazon CloudWatch Logs che riceve i dati dei parametri di monitoraggio avanzato per l'istanza database.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se l'autenticazione di Amazon Identity and Access Management (IAM) è abilitata, in caso contrario false.

- InstanceCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica data e ora di creazione dell'istanza database.

- Iops: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il valore di Provisioned IOPS (operazioni di I/O al secondo).

- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato: la crittografia per le istanze database viene gestita dal cluster di database.

- LatestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.

- LicenseModel: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Informazioni sul modello di licenza per questa istanza database.
- MonitoringInterval: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Intervallo, in secondi, tra i punti in cui vengono raccolti i parametri di monitoraggio avanzato per l'istanza database.
- MonitoringRoleArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN del ruolo IAM che permette a Neptune di inviare i parametri di monitoraggio avanzato ad Amazon CloudWatch Logs.
- MultiAZ: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se l'istanza database è un'implementazione Multi-AZ.
- PendingModifiedValues: un oggetto PendingModifiedValues.

  Specifica che le modifiche per l'istanza database sono in sospeso. Questo elemento è incluso solo quando sono presenti modifiche in sospeso. Le modifiche specifiche sono identificate dagli elementi secondari.
- PreferredBackupWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.
- PromotionTier: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Valore che specifica l'ordine di promozione di una replica di lettura a istanza primaria dopo un errore dell'istanza primaria esistente.
- PubliclyAccessible: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Questo flag non deve più essere utilizzato.
- ReadReplicaDBClusterIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori dei cluster di database che sono repliche di lettura di questa istanza database.

- ReadReplicaDBInstanceIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori delle repliche di lettura associate a questa istanza database.
- ReadReplicaSourceDBInstanceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene l'identificatore dell'istanza database di origine se questa istanza database è una replica di lettura.
- SecondaryAvailabilityZone: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se presente, specifica il nome della zona di disponibilità secondaria per un'istanza database con supporto per AZ multiple.
- StatusInfos: una matrice di oggetti DBInstanceStatusInfo.

  Stato di una replica di lettura. Se l'istanza non è una replica di lettura, questo campo è vuoto.
- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Non supportato: la crittografia per le istanze database viene gestita dal cluster di database.
- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il tipo di archiviazione associato all'istanza database.
- TdeCredentialArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN dell'archivio chiavi con cui l'istanza è associata per la crittografia TDE.
- Timezone: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato.
- VpcSecurityGroups: una matrice di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco di elementi del gruppo di sicurezza VPC a cui appartiene l'istanza database.

Errori

- DBInstanceNotFoundFault
- InvalidDBInstanceStateFault
- DBSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault

- InvalidDBClusterStateFault

ModifyDBInstance (operazione)

Il nome AWS CLI per questa API è: `modify-db-instance`.

Modifica le impostazioni di un'istanza database. Puoi modificare uno o più parametri di configurazione del database specificando questi parametri e i nuovi valori nella richiesta. Per informazioni sulle modifiche che puoi apportare all'istanza database, chiama the section called "DescribeValidDBInstanceModifications" prima di chiamare the section called "ModifyDBInstance".

Richiesta

- AllowMajorVersionUpgrade (nella CLI: `--allow-major-version-upgrade`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica che gli aggiornamenti delle versioni principali sono permessi. La modifica di questo parametro non comporta un'interruzione e la modifica viene applicata in modo asincrono il prima possibile.
- ApplyImmediately (nella CLI: `--apply-immediately`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se le modifiche in questa richiesta e le eventuali modifiche in sospeso vengono applicate in modo asincrono appena possibile, indipendentemente dall'impostazione di `PreferredMaintenanceWindow` per l'istanza database.

  Se questo parametro è impostato su `false`, le modifiche all'istanza database vengono applicate durante la finestra di manutenzione successiva. Le modifiche di alcuni parametri possono provocare un'interruzione e vengono applicate alla successiva chiamata a the section called "RebootDBInstance" oppure al successivo riavvio in seguito a un errore.

  Impostazione predefinita: `false`
- AutoMinorVersionUpgrade (nella CLI: `--auto-minor-version-upgrade`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Indica che gli aggiornamenti delle versioni secondarie vengono applicati automaticamente all'istanza database durante la finestra di manutenzione. La modifica di questo parametro non comporta un'interruzione, ad eccezione del caso seguente, e la modifica viene applicata in modo asincrono il prima possibile. Si verifica un'interruzione se questo parametro è impostato su `true`

durante la finestra di manutenzione ed è disponibile una versione secondaria più recente e Neptune ha abilitato l'applicazione di patch automatica per tale versione del motore.

- BackupRetentionPeriod (nella CLI: `--backup-retention-period`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Non applicabile. Il periodo di retention per i backup automatici viene gestito dal cluster di database. Per ulteriori informazioni, consulta the section called “ModifyDBCluster”.

  Impostazione predefinita: viene utilizzata l'impostazione esistente

- CACertificateIdentifier (nella CLI: `--ca-certificate-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica il certificato che deve essere associato all'istanza.

- CloudwatchLogsExportConfiguration (nella CLI: `--cloudwatch-logs-export-configuration`): un oggetto CloudwatchLogsExportConfiguration.

  L'impostazione di configurazione per i tipi di log per consentire l'esportazione in CloudWatch Logs per una determinata istanza di database o un determinato cluster di database.

- CopyTagsToSnapshot (nella CLI: `--copy-tags-to-snapshot`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  True per copiare tutti i tag dall'istanza database agli snapshot dell'istanza database, in caso contrario false. Il valore di default è false.

- DBInstanceClass (nella CLI: `--db-instance-class`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nuova capacità di calcolo e di memoria dell'istanza database, ad esempio `db.m4.large`. Non tutte le classi di istanza database sono disponibili in tutte le regioni Amazon.

  Se modifichi la classe dell'istanza database, durante la modifica si verifica un'interruzione. La modifica viene applicata durante la finestra di manutenzione successiva, a meno che il valore di `ApplyImmediately` per questa richiesta non sia `true`.

  Impostazione predefinita: viene utilizzata l'impostazione esistente

- DBInstanceIdentifier (nella CLI: `--db-instance-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore istanze DB. Questo valore è archiviato come stringa in caratteri minuscoli.

Vincoli:

- Deve corrispondere all'identificatore di un oggetto DBInstance esistente.

- DBParameterGroupName (nella CLI: `--db-parameter-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del gruppo di parametri database da applicare all'istanza database. La modifica di questa impostazione non comporta un'interruzione. Il nome del gruppo di parametri viene modificato immediatamente, ma le modifiche vere e proprie ai parametri vengono applicate solo al riavvio dell'istanza senza failover. L'istanza database NON verrà riavviata automaticamente e le modifiche ai parametri NON verranno applicate durante la finestra di manutenzione successiva.

  Impostazione predefinita: viene utilizzata l'impostazione esistente

  Vincoli: il gruppo di parametri database deve trovarsi nella stessa famiglia del gruppo di parametri database dell'istanza database.

- DBPortNumber (nella CLI: `--db-port-number`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Il numero della porta sulla quale il database accetta connessioni.

  Il valore del parametro `DBPortNumber` non deve corrispondere ad altri valori di porta specificati per le opzioni nel gruppo di opzioni per l'istanza database.

  Quando modifichi il valore di `DBPortNumber`, il database viene riavviato indipendentemente dal valore del parametro `ApplyImmediately`.

  Impostazione predefinita: `8182`

- DBSecurityGroups (nella CLI: `--db-security-groups`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco dei gruppi di sicurezza DB da autorizzare in questa istanza database. La modifica di questa impostazione non comporta un'interruzione e la modifica viene applicata in modo asincrono il prima possibile.

  Vincoli:

  - Se viene specificato, deve corrispondere a un oggetto DBSecurityGroups esistente.

- DBSubnetGroupName (nella CLI: `--db-subnet-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

Nuovo gruppo di sottoreti database per l'istanza database. Puoi utilizzare questo parametro per spostare l'istanza database in un VPC diverso.

Durante la modifica del gruppo di sottoreti si verifica un'interruzione. La modifica viene applicata durante la finestra di manutenzione successiva, a meno che non specifichi `true` per il parametro `ApplyImmediately`.

Vincoli: se specificato, deve corrispondere al nome di un oggetto DBSubnetGroup esistente.

Esempio: `mySubnetGroup`

- DeletionProtection (nella CLI: `--deletion-protection`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se per l'istanza database è abilitata la protezione da eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione. Per impostazione predefinita, la protezione da eliminazione è disabilitata. Consultare Eliminazione di un'istanza database.
- Domain (nella CLI: `--domain`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato.
- DomainIAMRoleName (nella CLI: `--domain-iam-role-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato
- EnableIAMDatabaseAuthentication (nella CLI: `--enable-iam-database-authentication`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  True per abilitare la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database, in caso contrario false.

  Puoi abilitare l'autenticazione del database IAM per i motori di database seguenti

  Non applicabile. La mappatura degli account Amazon IAM agli account di database viene gestita dal cluster database. Per ulteriori informazioni, consulta the section called “ModifyDBCluster”.

  Default: `false`
- EngineVersion (nella CLI: `--engine-version`): una stringa di tipo `string` (una stringa con codifica UTF-8).

Numero di versione del motore di database a cui eseguire l'aggiornamento. Attualmente, l'impostazione di questo parametro non ha alcun effetto. Per aggiornare il motore di database alla versione più recente, utilizzare l'API the section called “ApplyPendingMaintenanceAction”.

- Iops (nella CLI: `--iops`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Nuovo valore di Provisioned IOPS (operazioni di I/O al secondo) per l'istanza.

  La modifica di questa impostazione non comporta un'interruzione e la modifica viene applicata durante la finestra di manutenzione successiva, a meno che il parametro `ApplyImmediately` non sia impostato su `true` per questa richiesta.

  Impostazione predefinita: viene utilizzata l'impostazione esistente

- MonitoringInterval (nella CLI: `--monitoring-interval`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Intervallo, in secondi, tra i punti in cui vengono raccolti i parametri di monitoraggio avanzato per l'istanza database. Per disabilitare la raccolta dei parametri di monitoraggio avanzato, specifica 0. Il valore predefinito è 0.

  Se specifichi `MonitoringRoleArn`, devi anche impostare `MonitoringInterval` su un valore diverso da 0.

  Valori validi: `0, 1, 5, 10, 15, 30, 60`

- MonitoringRoleArn (nella CLI: `--monitoring-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN del ruolo IAM che permette a Neptune di inviare i parametri di monitoraggio avanzato ad Amazon CloudWatch Logs. Ad esempio, `arn:aws:iam:123456789012:role/emaccess`.

  Se `MonitoringInterval` è impostato su un valore diverso da 0, devi fornire un valore di `MonitoringRoleArn`.

- MultiAZ (nella CLI: `--multi-az`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se l'istanza database è un'implementazione Multi-AZ. La modifica di questo parametro non comporta un'interruzione e la modifica viene applicata durante la finestra di manutenzione

successiva, a meno che il parametro `ApplyImmediately` non sia impostato su `true` per questa richiesta.

- NewDBInstanceIdentifier (nella CLI: `--new-db-instance-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nuovo identificatore dell'istanza database quando un'istanza database viene rinominata. Quando modifichi l'identificatore istanze DB, l'istanza viene riavviata immediatamente se imposti `Apply Immediately` su true oppure il riavvio viene eseguito durante la finestra di manutenzione successiva se `Apply Immediately` è false. Questo valore è archiviato come stringa in caratteri minuscoli.

  Vincoli:

  - Deve contenere da 1 a 63 lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.

  Esempio: `mydbinstance`
- PreferredBackupWindow (nella CLI: `--preferred-backup-window`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Intervallo di tempo giornaliero durante il quale vengono creati i backup automatici, se abilitati.

  Non applicabile. L'intervallo di tempo giornaliero per la creazione dei backup automatici viene gestito dal cluster di database. Per ulteriori informazioni, consulta the section called “ModifyDBCluster”.

  Vincoli:

  - Il valore deve essere nel formato hh24:mi-hh24:mi
  - Il valore deve essere nel fuso orario UTC (Universal Coordinated Time)
  - Il valore non deve essere in conflitto con la finestra di manutenzione preferita
  - Il valore deve essere almeno di 30 minuti
- PreferredMaintenanceWindow (nella CLI: `--preferred-maintenance-window`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Intervallo di tempo settimanale (nel fuso orario UTC) durante il quale può verificarsi la finestra di manutenzione del sistema, che potrebbe comportare un'interruzione. La modifica di questo parametro non comporta un'interruzione, ad eccezione della situazione seguente, e la modifica

viene applicata in modo asincrono il prima possibile. Se sono presenti operazioni in sospeso che provocano un riavvio e la finestra di manutenzione viene modificata per includere l'ora corrente, la modifica di questo parametro causa il riavvio dell'istanza database. Se sposti questa finestra sull'ora corrente, vi devono essere almeno 30 minuti tra l'ora corrente e la fine della finestra per assicurare che le modifiche in sospeso vengano applicate.

Impostazione predefinita: viene utilizzata l'impostazione esistente

Formato: ddd:hh24:mi-ddd:hh24:mi

Giorni validi: lunedì, martedì, mercoledì, giovedì, venerdì, sabato, domenica

Vincoli: il valore deve essere almeno di 30 minuti

- PromotionTier (nella CLI: `--promotion-tier`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Valore che specifica l'ordine di promozione di una replica di lettura a istanza primaria dopo un errore dell'istanza primaria esistente.

  Impostazione predefinita: 1

  Valori validi: 0-15

- PubliclyAccessible (nella CLI: `--publicly-accessible`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Questo flag non deve più essere utilizzato.

- StorageType (nella CLI: `--storage-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non applicabile. In Neptune il tipo di archiviazione è gestito a livello di cluster di database.

- TdeCredentialArn (nella CLI: `--tde-credential-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN dell'archivio chiavi con cui associare l'istanza per la crittografia TDE.

- TdeCredentialPassword (nella CLI: `--tde-credential-password`): una SensitiveString di tipo `string` (una stringa con codifica UTF-8).

  Password per l'ARN specificato dall'archivio chiavi per accedere al dispositivo.

- VpcSecurityGroupIds (nella CLI: `--vpc-security-group-ids`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco dei gruppi di sicurezza VPC EC2 da autorizzare in questa istanza database. Questa modifica viene applicata in modo asincrono il prima possibile.

  Non applicabile. L'elenco associato di gruppi di sicurezza VPC EC2 viene gestito dal cluster di database. Per ulteriori informazioni, consulta the section called “ModifyDBCluster”.

  Vincoli:

  - Se viene specificato, deve corrispondere a un oggetto VpcSecurityGroupIds esistente.

Risposta

Contiene i dettagli di un'istanza database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called “DescribeDBInstances”.

- AutoMinorVersionUpgrade: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica che le patch della versione secondaria vengono applicate automaticamente.
- AvailabilityZone: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome della zona di disponibilità in cui si trova l'istanza database.
- BackupRetentionPeriod: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.
- CACertificateIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore del certificato CA per questa istanza database.
- CopyTagsToSnapshot: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se i tag vengono copiati dall'istanza database agli snapshot dell'istanza database.
- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se l'istanza database è membro di un cluster di database, contiene il nome del cluster di database di cui l'istanza database è membro.
- DBInstanceArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

Amazon Resource Name (ARN) per l'istanza database.

- DBInstanceClass: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nome della classe di capacità di calcolo e di memoria dell'istanza database.

- DBInstanceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un'istanza database.

- DbInstancePort: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto l'istanza database. Se l'istanza database fa parte di un cluster di database, questa porta può essere diversa rispetto a quella del cluster di database.

- DBInstanceStatus: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente di questo database.

- DbiResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore univoco e non modificabile della regione Amazon per l'istanza database. Questo identificatore è reperibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per l'istanza database.

- DBName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del database.

- DBParameterGroups: una matrice di oggetti DBParameterGroupStatus.

  Fornisce l'elenco dei gruppi di parametri database applicati a questa istanza database.

- DBSecurityGroups: una matrice di oggetti DBSecurityGroupMembership.

  Fornisce l'elenco degli elementi del gruppo di sicurezza DB contenenti solo elementi secondari `DBSecurityGroup.Name` e `DBSecurityGroup.Status`.

- DBSubnetGroup: un oggetto DBSubnetGroup.

  Specifica le informazioni sul gruppo di sottoreti associato all'istanza database, inclusi il nome, la descrizione e le sottoreti presenti nel gruppo.

- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

Un valore che indica se per l'istanza database è abilitata o meno la protezione da eliminazione. L'istanza non può essere eliminata se è abilitata la protezione da eliminazione. Consultare Eliminazione di un'istanza database.

- DomainMemberships: una matrice di oggetti DomainMembership.

  Non supportato

- EnabledCloudwatchLogsExports: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questa istanza database è configurata per esportare in CloudWatch Logs.

- Endpoint: un oggetto Endpoint.

  Specifica l'endpoint di connessione.

- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questa istanza database.

- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.

- EnhancedMonitoringResourceArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) del flusso di log Amazon CloudWatch Logs che riceve i dati dei parametri di monitoraggio avanzato per l'istanza database.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se l'autenticazione di Amazon Identity and Access Management (IAM) è abilitata, in caso contrario false.

- InstanceCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica data e ora di creazione dell'istanza database.

- Iops: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il valore di Provisioned IOPS (operazioni di I/O al secondo).

- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato: la crittografia per le istanze database viene gestita dal cluster di database.

- LatestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.
- LicenseModel: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Informazioni sul modello di licenza per questa istanza database.
- MonitoringInterval: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Intervallo, in secondi, tra i punti in cui vengono raccolti i parametri di monitoraggio avanzato per l'istanza database.
- MonitoringRoleArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN del ruolo IAM che permette a Neptune di inviare i parametri di monitoraggio avanzato ad Amazon CloudWatch Logs.
- MultiAZ: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se l'istanza database è un'implementazione Multi-AZ.
- PendingModifiedValues: un oggetto PendingModifiedValues.

  Specifica che le modifiche per l'istanza database sono in sospeso. Questo elemento è incluso solo quando sono presenti modifiche in sospeso. Le modifiche specifiche sono identificate dagli elementi secondari.
- PreferredBackupWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.
- PromotionTier: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Valore che specifica l'ordine di promozione di una replica di lettura a istanza primaria dopo un errore dell'istanza primaria esistente.
- PubliclyAccessible: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Questo flag non deve più essere utilizzato.

- ReadReplicaDBClusterIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori dei cluster di database che sono repliche di lettura di questa istanza database.
- ReadReplicaDBInstanceIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori delle repliche di lettura associate a questa istanza database.
- ReadReplicaSourceDBInstanceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene l'identificatore dell'istanza database di origine se questa istanza database è una replica di lettura.
- SecondaryAvailabilityZone: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se presente, specifica il nome della zona di disponibilità secondaria per un'istanza database con supporto per AZ multiple.
- StatusInfos: una matrice di oggetti DBInstanceStatusInfo.

  Stato di una replica di lettura. Se l'istanza non è una replica di lettura, questo campo è vuoto.
- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Non supportato: la crittografia per le istanze database viene gestita dal cluster di database.
- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il tipo di archiviazione associato all'istanza database.
- TdeCredentialArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN dell'archivio chiavi con cui l'istanza è associata per la crittografia TDE.
- Timezone: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato.
- VpcSecurityGroups: una matrice di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco di elementi del gruppo di sicurezza VPC a cui appartiene l'istanza database.

Errori

- InvalidDBInstanceStateFault

- InvalidDBSecurityGroupStateFault
- DBInstanceAlreadyExistsFault
- DBInstanceNotFoundFault
- DBSecurityGroupNotFoundFault
- DBParameterGroupNotFoundFault
- InsufficientDBInstanceCapacityFault
- StorageQuotaExceededFault
- InvalidVPCNetworkStateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBUpgradeDependencyFailureFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- CertificateNotFoundFault
- DomainNotFoundFault

RebootDBInstance (operazione)

Il nome AWS CLI per questa API è: `reboot-db-instance`.

Potrebbe essere necessario riavviare l'istanza database, in genere per motivi di manutenzione. Se, ad esempio, apporti determinate modifiche oppure se modifichi il gruppo di parametri database associato all'istanza database, è necessario riavviare l'istanza affinché le modifiche vengano applicate.

Il riavvio di un'istanza database comporta il riavvio del servizio del motore di database. Il riavvio di un'istanza database comporta un'interruzione temporanea, durante la quale lo stato dell'istanza database viene impostato su rebooting (riavvio in corso).

Richiesta

- DBInstanceIdentifier (nella CLI: `--db-instance-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore istanze DB. Questo parametro è archiviato come stringa in minuscolo.

Vincoli:

- Deve corrispondere all'identificatore di un oggetto DBInstance esistente.

- ForceFailover (nella CLI: `--force-failover`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se il valore è `true`, il riavvio viene eseguito tramite un failover AZ multiple.

  Vincolo: non è possibile specificare `true` se l'istanza non è configurata per AZ multiple.

Risposta

Contiene i dettagli di un'istanza database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called "DescribeDBInstances".

- AutoMinorVersionUpgrade: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica che le patch della versione secondaria vengono applicate automaticamente.

- AvailabilityZone: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome della zona di disponibilità in cui si trova l'istanza database.

- BackupRetentionPeriod: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.

- CACertificateIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore del certificato CA per questa istanza database.

- CopyTagsToSnapshot: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se i tag vengono copiati dall'istanza database agli snapshot dell'istanza database.

- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se l'istanza database è membro di un cluster di database, contiene il nome del cluster di database di cui l'istanza database è membro.

- DBInstanceArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per l'istanza database.

- DBInstanceClass: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nome della classe di capacità di calcolo e di memoria dell'istanza database.
- DBInstanceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un'istanza database.
- DbInstancePort: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto l'istanza database. Se l'istanza database fa parte di un cluster di database, questa porta può essere diversa rispetto a quella del cluster di database.
- DBInstanceStatus: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente di questo database.
- DbiResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore univoco e non modificabile della regione Amazon per l'istanza database. Questo identificatore è reperibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per l'istanza database.
- DBName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del database.
- DBParameterGroups: una matrice di oggetti DBParameterGroupStatus.

  Fornisce l'elenco dei gruppi di parametri database applicati a questa istanza database.
- DBSecurityGroups: una matrice di oggetti DBSecurityGroupMembership.

  Fornisce l'elenco degli elementi del gruppo di sicurezza DB contenenti solo elementi secondari `DBSecurityGroup.Name` e `DBSecurityGroup.Status`.
- DBSubnetGroup: un oggetto DBSubnetGroup.

  Specifica le informazioni sul gruppo di sottoreti associato all'istanza database, inclusi il nome, la descrizione e le sottoreti presenti nel gruppo.
- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se per l'istanza database è abilitata o meno la protezione da eliminazione. L'istanza non può essere eliminata se è abilitata la protezione da eliminazione. Consultare Eliminazione di un'istanza database.

- DomainMemberships: una matrice di oggetti DomainMembership.

  Non supportato

- EnabledCloudwatchLogsExports: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questa istanza database è configurata per esportare in CloudWatch Logs.

- Endpoint: un oggetto Endpoint.

  Specifica l'endpoint di connessione.

- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questa istanza database.

- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.

- EnhancedMonitoringResourceArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) del flusso di log Amazon CloudWatch Logs che riceve i dati dei parametri di monitoraggio avanzato per l'istanza database.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se l'autenticazione di Amazon Identity and Access Management (IAM) è abilitata, in caso contrario false.

- InstanceCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica data e ora di creazione dell'istanza database.

- Iops: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il valore di Provisioned IOPS (operazioni di I/O al secondo).

- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato: la crittografia per le istanze database viene gestita dal cluster di database.

- LatestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.

- LicenseModel: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Informazioni sul modello di licenza per questa istanza database.

- MonitoringInterval: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Intervallo, in secondi, tra i punti in cui vengono raccolti i parametri di monitoraggio avanzato per l'istanza database.

- MonitoringRoleArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN del ruolo IAM che permette a Neptune di inviare i parametri di monitoraggio avanzato ad Amazon CloudWatch Logs.

- MultiAZ: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se l'istanza database è un'implementazione Multi-AZ.

- PendingModifiedValues: un oggetto PendingModifiedValues.

  Specifica che le modifiche per l'istanza database sono in sospeso. Questo elemento è incluso solo quando sono presenti modifiche in sospeso. Le modifiche specifiche sono identificate dagli elementi secondari.

- PreferredBackupWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.

- PromotionTier: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Valore che specifica l'ordine di promozione di una replica di lettura a istanza primaria dopo un errore dell'istanza primaria esistente.

- PubliclyAccessible: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Questo flag non deve più essere utilizzato.

- ReadReplicaDBClusterIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori dei cluster di database che sono repliche di lettura di questa istanza database.

- ReadReplicaDBInstanceIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori delle repliche di lettura associate a questa istanza database.
- ReadReplicaSourceDBInstanceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene l'identificatore dell'istanza database di origine se questa istanza database è una replica di lettura.
- SecondaryAvailabilityZone: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se presente, specifica il nome della zona di disponibilità secondaria per un'istanza database con supporto per AZ multiple.
- StatusInfos: una matrice di oggetti DBInstanceStatusInfo.

  Stato di una replica di lettura. Se l'istanza non è una replica di lettura, questo campo è vuoto.
- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Non supportato: la crittografia per le istanze database viene gestita dal cluster di database.
- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il tipo di archiviazione associato all'istanza database.
- TdeCredentialArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN dell'archivio chiavi con cui l'istanza è associata per la crittografia TDE.
- Timezone: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato.
- VpcSecurityGroups: una matrice di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco di elementi del gruppo di sicurezza VPC a cui appartiene l'istanza database.

Errori

- InvalidDBInstanceStateFault
- DBInstanceNotFoundFault

DescribeDBInstances (operazione)

Il nome AWS CLI per questa API è: `describe-db-instances`.

Restituisce informazioni sulle istanze di provisioning e supporta l'impaginazione.

> **Note**
>
> Questa operazione può anche restituire informazioni per le istanze di Amazon RDS e le istanze di Amazon DocDB.

Richiesta

- DBInstanceIdentifier (nella CLI: `--db-instance-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore dell'istanza fornito dall'utente. Se questo parametro viene specificato, vengono restituite solo le informazioni dell'istanza database specifica. Questo parametro non fa distinzione tra maiuscole e minuscole.

  Vincoli:

  - Se viene specificato, deve corrispondere all'identificatore di un oggetto DBInstance esistente.
- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Filtro che specifica una o più istanze database da descrivere.

  Filtri supportati:

  - `db-cluster-id`: accetta identificatori e Amazon Resource Name (ARN) di cluster di database. L'elenco di risultati includerà solo le informazioni sulle istanze database associate ai cluster di database identificati da questi ARN.
  - `engine`: accetta un nome del motore (ad esempio `neptune`) e limita l'elenco dei risultati alle istanze database create da tale motore.

  Ad esempio, per richiamare questa API dall'interfaccia della riga di comando Amazon e filtrare in modo che vengano restituite solo le istanze database Neptune, potresti utilizzare il seguente comando:

Example

```
aws neptune describe-db-instances \
            --filters  Name=engine,Values=neptune
```

- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta `DescribeDBInstances` precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.

Risposta

- DBInstances: una matrice di oggetti DBInstance.

  Elenco di istanze di the section called “DBInstance”.

- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

Errori

- DBInstanceNotFoundFault

DescribeOrderableDBInstanceOptions (operazione)

Il nome AWS CLI per questa API è: `describe-orderable-db-instance-options`.

Restituisce un elenco delle opzioni delle istanze database ordinabili per il motore specificato.

Richiesta

- DBInstanceClass (nella CLI: `--db-instance-class`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Valore di filtro della classe di istanza database. Specifica questo parametro per visualizzare solo le offerte disponibili per la classe di istanza database specificata.
- Engine (nella CLI: `--engine`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del motore per cui recuperare le opzioni delle istanze database.
- EngineVersion (nella CLI: `--engine-version`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Valore di filtro della versione del motore. Specifica questo parametro per visualizzare solo le offerte disponibili per la versione del motore specificata.
- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Questo parametro non è attualmente supportato.
- LicenseModel (nella CLI: `--license-model`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Valore di filtro del modello di licenza. Specifica questo parametro per visualizzare solo le offerte disponibili per il modello di licenza specificato.
- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta DescribeOrderableDBInstanceOptions precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.
- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

Impostazione predefinita: 100

Vincoli: minimo 20, massimo 100.

- Vpc (nella CLI: `--vpc`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Valore di filtro VPC. Specifica questo parametro per visualizzare solo le offerte VPC o non VPC disponibili.

Risposta

- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta OrderableDBInstanceOptions precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.
- OrderableDBInstanceOptions: una matrice di oggetti OrderableDBInstanceOption.

  Struttura the section called “OrderableDBInstanceOption” contenente le informazioni sulle opzioni ordinabili per l'istanza database.

DescribeValidDBInstanceModifications (operazione)

Il nome AWS CLI per questa API è: `describe-valid-db-instance-modifications`.

Puoi chiamare the section called “DescribeValidDBInstanceModifications” per informazioni sulle modifiche che puoi apportare all'istanza database. Puoi usare queste informazioni quando chiami the section called “ModifyDBInstance”.

Richiesta

- DBInstanceIdentifier (nella CLI: `--db-instance-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

Identificatore del cliente o ARN dell'istanza database.

Risposta

Informazioni sulle modifiche valide che è possibile apportare all'istanza database. Contiene il risultato di una chiamata riuscita all'operazione the section called “DescribeValidDBInstanceModifications”. Puoi usare queste informazioni quando chiami the section called “ModifyDBInstance”.

- Storage: una matrice di oggetti ValidStorageOptions.

  Opzioni di storage valide per l'istanza database.

Errori

- DBInstanceNotFoundFault
- InvalidDBInstanceStateFault

Strutture:

DBInstance (struttura)

Contiene i dettagli di un'istanza database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called “DescribeDBInstances”.

Campi

- AutoMinorVersionUpgrade: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica che le patch della versione secondaria vengono applicate automaticamente.
- AvailabilityZone: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome della zona di disponibilità in cui si trova l'istanza database.
- BackupRetentionPeriod: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.

- CACertificateIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore del certificato CA per questa istanza database.

- CopyTagsToSnapshot: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se i tag vengono copiati dall'istanza database agli snapshot dell'istanza database.

- DBClusterIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se l'istanza database è membro di un cluster di database, contiene il nome del cluster di database di cui l'istanza database è membro.

- DBInstanceArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per l'istanza database.

- DBInstanceClass: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nome della classe di capacità di calcolo e di memoria dell'istanza database.

- DBInstanceIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un'istanza database.

- DbInstancePort: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto l'istanza database. Se l'istanza database fa parte di un cluster di database, questa porta può essere diversa rispetto a quella del cluster di database.

- DBInstanceStatus: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente di questo database.

- DbiResourceId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore univoco e non modificabile della regione Amazon per l'istanza database. Questo identificatore è reperibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per l'istanza database.

- DBName: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del database.

- DBParameterGroups: questo è un array di oggetti DBParameterGroupStatus.

  Fornisce l'elenco dei gruppi di parametri database applicati a questa istanza database.
- DBSecurityGroups: questo è un array di oggetti DBSecurityGroupMembership.

  Fornisce l'elenco degli elementi del gruppo di sicurezza DB contenenti solo elementi secondari `DBSecurityGroup.Name` e `DBSecurityGroup.Status`.
- DBSubnetGroup: questo è un oggetto DBSubnetGroup.

  Specifica le informazioni sul gruppo di sottoreti associato all'istanza database, inclusi il nome, la descrizione e le sottoreti presenti nel gruppo.
- DeletionProtection: questo è un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se per l'istanza database è abilitata o meno la protezione da eliminazione. L'istanza non può essere eliminata se è abilitata la protezione da eliminazione. Consultare Eliminazione di un'istanza database.
- DomainMemberships: questo è un array di oggetti DomainMembership.

  Non supportato
- EnabledCloudwatchLogsExports: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questa istanza database è configurata per esportare in CloudWatch Logs.
- Endpoint: questo è un oggetto Endpoint.

  Specifica l'endpoint di connessione.
- Engine: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questa istanza database.
- EngineVersion: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.
- EnhancedMonitoringResourceArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) del flusso di log Amazon CloudWatch Logs che riceve i dati dei parametri di monitoraggio avanzato per l'istanza database.

- IAMDatabaseAuthenticationEnabled: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se l'autenticazione di Amazon Identity and Access Management (IAM) è abilitata, in caso contrario false.

- InstanceCreateTime: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica data e ora di creazione dell'istanza database.

- Iops: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il valore di Provisioned IOPS (operazioni di I/O al secondo).

- KmsKeyId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato: la crittografia per le istanze database viene gestita dal cluster di database.

- LatestRestorableTime: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.

- LicenseModel: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Informazioni sul modello di licenza per questa istanza database.

- MonitoringInterval: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Intervallo, in secondi, tra i punti in cui vengono raccolti i parametri di monitoraggio avanzato per l'istanza database.

- MonitoringRoleArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN del ruolo IAM che permette a Neptune di inviare i parametri di monitoraggio avanzato ad Amazon CloudWatch Logs.

- MultiAZ: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se l'istanza database è un'implementazione Multi-AZ.

- PendingModifiedValues: questo è un oggetto PendingModifiedValues.

Specifica che le modifiche per l'istanza database sono in sospeso. Questo elemento è incluso solo quando sono presenti modifiche in sospeso. Le modifiche specifiche sono identificate dagli elementi secondari.

- PreferredBackupWindow: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.

- PromotionTier: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Valore che specifica l'ordine di promozione di una replica di lettura a istanza primaria dopo un errore dell'istanza primaria esistente.

- PubliclyAccessible: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Questo flag non deve più essere utilizzato.

- ReadReplicaDBClusterIdentifiers: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori dei cluster di database che sono repliche di lettura di questa istanza database.

- ReadReplicaDBInstanceIdentifiers: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori delle repliche di lettura associate a questa istanza database.

- ReadReplicaSourceDBInstanceIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene l'identificatore dell'istanza database di origine se questa istanza database è una replica di lettura.

- SecondaryAvailabilityZone: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

Se presente, specifica il nome della zona di disponibilità secondaria per un'istanza database con supporto per AZ multiple.

- StatusInfos: questo è un array di oggetti DBInstanceStatusInfo.

  Stato di una replica di lettura. Se l'istanza non è una replica di lettura, questo campo è vuoto.

- StorageEncrypted: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Non supportato: la crittografia per le istanze database viene gestita dal cluster di database.

- StorageType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il tipo di archiviazione associato all'istanza database.

- TdeCredentialArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  ARN dell'archivio chiavi con cui l'istanza è associata per la crittografia TDE.

- Timezone: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato.

- VpcSecurityGroups: questo è un array di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco di elementi del gruppo di sicurezza VPC a cui appartiene l'istanza database.

`DBInstance` viene utilizzato come elemento di risposta per:

- CreateDBInstance
- DeleteDBInstance
- ModifyDBInstance
- RebootDBInstance

DBInstanceStatusInfo (struttura)

Fornisce un elenco di informazioni sullo stato per un'istanza database.

Campi

- Message: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

Dettagli dell'errore in caso di errore per l'istanza. Se l'istanza non è in stato di errore, il valore è vuoto.

- Normal: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Valore booleano corrispondente a true se l'istanza funziona normalmente oppure false se l'istanza è in stato di errore.
- Status: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Stato dell'istanza database. I valori di StatusType per una replica di lettura possono essere replicating, error, stopped o terminated.
- StatusType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Questo valore indica attualmente una "replica di lettura".

OrderableDBInstanceOption (struttura)

Contiene un elenco di opzioni disponibili per un'istanza database.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called “DescribeOrderableDBInstanceOptions”.

Campi

- AvailabilityZones: questo è un array di oggetti AvailabilityZone.

  Elenco di zone di disponibilità per un'istanza database.
- DBInstanceClass: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Classe di un'istanza database.
- Engine: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Tipo di motore di un'istanza database.
- EngineVersion: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Versione del motore di un'istanza database.
- LicenseModel: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Modello di licenza per un'istanza database.

- MaxIopsPerDbInstance: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Valore massimo totale di Provisioned IOPS per un'istanza database.
- MaxIopsPerGib: questo è un valore DoubleOptional di tipo `double` (un numero a virgola mobile IEEE 754 a doppia precisione).

  Valore massimo di Provisioned IOPS per GiB per un'istanza database.
- MaxStorageSize: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Dimensioni di storage massime per un'istanza database.
- MinIopsPerDbInstance: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Valore minimo totale di Provisioned IOPS per un'istanza database.
- MinIopsPerGib: questo è un valore DoubleOptional di tipo `double` (un numero a virgola mobile IEEE 754 a doppia precisione).

  Valore minimo di Provisioned IOPS per GiB per un'istanza database.
- MinStorageSize: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Dimensioni di storage minime per un'istanza database.
- MultiAZCapable: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se un'istanza database fornisce supporto per AZ multiple.
- ReadReplicaCapable: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se un'istanza database può avere una replica di lettura.
- StorageType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non applicabile. In Neptune il tipo di archiviazione è gestito a livello di cluster di database.
- SupportsEnhancedMonitoring: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se un'istanza database supporta il monitoraggio avanzato a intervalli da 1 a 60 secondi.

- SupportsGlobalDatabases: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se è possibile utilizzare i database globali Neptune con una combinazione specifica di altri attributi del motore di database.

- SupportsIAMDatabaseAuthentication: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se un'istanza database supporta l'autenticazione dei database IAM.

- SupportsIops: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se un'istanza database supporta l'opzione Provisioned IOPS.

- SupportsStorageEncryption: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se un'istanza database supporta lo storage crittografato.

- Vpc: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se un'istanza database si trova in un VPC.

PendingModifiedValues (struttura)

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called “ModifyDBInstance”.

Campi

- AllocatedStorage: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Contiene le nuove dimensioni di `AllocatedStorage` per l'istanza database che verrà applicata o è in fase di applicazione.

- BackupRetentionPeriod: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il numero di giorni in sospeso durante i quali vengono conservati i backup automatici.

- CACertificateIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

Specifica l'identificatore del certificato CA per l'istanza database.

- DBInstanceClass: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nuovo `DBInstanceClass` per l'istanza database che verrà applicata o è in fase di applicazione.

- DBInstanceIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nuovo `DBInstanceIdentifier` per l'istanza database che verrà applicata o è in fase di applicazione.

- DBSubnetGroupName: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nuovo gruppo di sottoreti database per l'istanza database.

- EngineVersion: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.

- Iops: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il nuova valore Provisioned IOPS per l'istanza database che verrà applicata o attualmente è in fase di applicazione.

- MultiAZ: questo è un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Indica che l'istanza database Single-AZ deve essere modificata in un'implementazione Multi-AZ.

- PendingCloudwatchLogsExports: questo è un oggetto PendingCloudwatchLogsExports.

  Questa struttura `PendingCloudwatchLogsExports` specifica su quali log di CloudWatch sono abilitate e disabilitate le modifiche in sospeso.

- Port: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la porta in sospeso per l'istanza database.

- StorageType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non applicabile. In Neptune il tipo di archiviazione è gestito a livello di cluster di database.

ValidStorageOptions (struttura)

Non applicabile. In Neptune il tipo di archiviazione è gestito a livello di cluster di database.

Campi

- IopsToStorageRatio: questo è un array di oggetti DoubleRange.

  Non applicabile. In Neptune il tipo di archiviazione è gestito a livello di cluster di database.
- ProvisionedIops: questo è un array di oggetti Intervallo.

  Non applicabile. In Neptune il tipo di archiviazione è gestito a livello di cluster di database.
- StorageSize: questo è un array di oggetti Intervallo.

  Non applicabile. In Neptune il tipo di archiviazione è gestito a livello di cluster di database.
- StorageType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non applicabile. In Neptune il tipo di archiviazione è gestito a livello di cluster di database.

ValidDBInstanceModificationsMessage (struttura)

Informazioni sulle modifiche valide che è possibile apportare all'istanza database. Contiene il risultato di una chiamata riuscita all'operazione the section called “DescribeValidDBInstanceModifications”. Puoi usare queste informazioni quando chiami the section called “ModifyDBInstance”.

Campi

- Storage: questo è un array di oggetti ValidStorageOptions.

  Opzioni di storage valide per l'istanza database.

`ValidDBInstanceModificationsMessage` viene utilizzato come elemento di risposta per:

- DescribeValidDBInstanceModifications

API dei parametri Neptune

Operazioni:

- CopyDBParameterGroup (operazione)
- CopyDBClusterParameterGroup (operazione)
- CreateDBParameterGroup (operazione)

- CreateDBClusterParameterGroup (operazione)
- DeleteDBParameterGroup (operazione)
- DeleteDBClusterParameterGroup (operazione)
- ModifyDBParameterGroup (operazione)
- ModifyDBClusterParameterGroup (operazione)
- ResetDBParameterGroup (operazione)
- ResetDBClusterParameterGroup (operazione)
- DescribeDBParameters (operazione)
- DescribeDBParameterGroups (operazione)
- DescribeDBClusterParameters (operazione)
- DescribeDBClusterParameterGroups (operazione)
- DescribeEngineDefaultParameters (operazione)
- DescribeEngineDefaultClusterParameters (operazione)

Strutture:

- Parametro (struttura)
- DBParameterGroup (struttura)
- DBClusterParameterGroup (struttura)
- DBParameterGroupStatus (struttura)

CopyDBParameterGroup (operazione)

Il nome AWS CLI per questa API è: `copy-db-parameter-group`.

Copia il gruppo di parametri database specificato.

Richiesta

- SourceDBParameterGroupIdentifier (nella CLI: `--source-db-parameter-group-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore o l'ARN per il gruppo di parametri database di origine. Per informazioni su come creare un ARN, consultare Costruzione di un ARN (Amazon Resource Name).

Vincoli:

- È necessario specificare un gruppo di parametri database valido.
- È necessario specificare un identificatore di gruppo di parametri database valido, ad esempio `my-db-param-group` o un ARN valido.

- Tags (nella CLI: `--tags`): un array di oggetti Tag.

  I tag da assegnare al gruppo di parametri database copiato.

- TargetDBParameterGroupDescription (nella CLI: `--target-db-parameter-group-description`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Una descrizione del gruppo di parametri database copiato.

- TargetDBParameterGroupIdentifier (nella CLI: `--target-db-parameter-group-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore del gruppo di parametri database copiato.

  Vincoli:

  - Non può essere null o vuoto.
  - Deve contenere da 1 a 255 lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.

  Esempio: `my-db-parameter-group`

Risposta

Contiene i dettagli di un gruppo di parametri database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called "DescribeDBParameterGroups".

- DBParameterGroupArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per il gruppo di parametri database.

- DBParameterGroupFamily: una stringa di tipo `string` (una stringa con codifica UTF-8).

Fornisce il nome della famiglia del gruppo di parametri database con la quale è compatibile questo gruppo di parametri database.

- DBParameterGroupName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del gruppo di parametri database.

- Description: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce la descrizione specificata dall'utente per questo gruppo di parametri database.

Errori

- DBParameterGroupNotFoundFault
- DBParameterGroupAlreadyExistsFault
- DBParameterGroupQuotaExceededFault

CopyDBClusterParameterGroup (operazione)

Il nome AWS CLI per questa API è: `copy-db-cluster-parameter-group`.

Copia il gruppo di parametri database cluster specificato.

Richiesta

- SourceDBClusterParameterGroupIdentifier (nella CLI: `--source-db-cluster-parameter-group-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore o l'Amazon Resource Name (ARN) del gruppo di parametri cluster di database di origine. Per informazioni su come creare un ARN, consultare Costruzione di un ARN (Amazon Resource Name).

  Vincoli:

  - È necessario specificare un gruppo di parametri database cluster valido.
  - Se il gruppo di parametri di cluster database di origine si trova nella stessa regione Amazon della copia, specifica un identificatore del gruppo di parametri database valido, ad esempio `my-db-cluster-param-group` o un ARN valido.

  - Se il gruppo di parametri database di origine si trova in una regione Amazon diversa rispetto alla copia, specifica un ARN del gruppo di parametri del cluster database valido, ad esempio `arn:aws:rds:us-east-1:123456789012:cluster-pg:custom-cluster-group1`.
- Tags (nella CLI: `--tags`): un array di oggetti Tag.

  I tag da assegnare al gruppo di parametri del cluster di database copiato.
- TargetDBClusterParameterGroupDescription (nella CLI: `--target-db-cluster-parameter-group-description`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Una descrizione del gruppo di parametri del cluster di database copiato.
- TargetDBClusterParameterGroupIdentifier (nella CLI: `--target-db-cluster-parameter-group-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore del gruppo di parametri database cluster copiato.

  Vincoli:
  - Non può essere null o vuoto
  - Deve contenere da 1 a 255 lettere, numeri o trattini
  - Il primo carattere deve essere una lettera
  - Non può terminare con un trattino o contenere due trattini consecutivi

  Esempio: `my-cluster-param-group1`

Risposta

Contiene i dettagli di un gruppo di parametri del cluster database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called "DescribeDBClusterParameterGroups".

- DBClusterParameterGroupArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per il gruppo di parametri del cluster database.
- DBClusterParameterGroupName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del gruppo di parametri del cluster database.
- DBParameterGroupFamily: una stringa di tipo `string` (una stringa con codifica UTF-8).

Fornisce il nome della famiglia del gruppo di parametri database con cui è compatibile questo gruppo di parametri del cluster database.

- Description: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce la descrizione specificata dall'utente per questo gruppo di parametri del cluster database.

Errori

- DBParameterGroupNotFoundFault
- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

CreateDBParameterGroup (operazione)

Il nome AWS CLI per questa API è: `create-db-parameter-group`.

Crea un nuovo gruppo di parametri database.

Un gruppo di parametri database viene creato inizialmente con i parametri predefiniti per il motore di database utilizzato dall'istanza database. Per offrire i valori personalizzati per uno qualsiasi dei parametri, è necessario modificare il gruppo dopo averlo creato utilizzando ModifyDBParameterGroup. Dopo aver creato un gruppo di parametri database, è necessario associarlo a un'istanza database utilizzando ModifyDBInstance. Quando si associa un nuovo gruppo di parametri database a un'istanza database in esecuzione, è necessario riavviare l'istanza database senza failover affinché il nuovo gruppo di parametri database e le relative impostazioni diventino effettivi.

Important

Dopo aver creato un gruppo di parametri database, è necessario attendere almeno 5 minuti prima di creare la prima istanza database che utilizza il gruppo di parametri database come gruppo predefinito. In questo modo, Amazon Neptune può completare la creazione dell'operazione prima che il gruppo di parametri venga utilizzato come predefinito per una nuova istanza database. Questo è particolarmente importante per parametri critici durante la creazione del database predefinito per un'istanza database, ad esempio il set di caratteri per il database predefinito specificato dal parametro `character_set_database`. È possibile utilizzare l'opzione Gruppi di parametri della console Amazon Neptune o il comando

DescribeDBParameters per verificare che il gruppo di parametri database sia stato creato o modificato.

Richiesta

- DBParameterGroupFamily (nella CLI: `--db-parameter-group-family`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della famiglia del gruppo di parametri di database. Un gruppo di parametri database può essere associato solo e unicamente a una famiglia di gruppo di parametri database e può essere applicato solo a un'istanza database che esegue un motore di database e una versione del motore compatibile con tale famiglia di gruppo di parametri database.
- DBParameterGroupName (nella CLI: `--db-parameter-group-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di parametri database.

  Vincoli:

  - Deve contenere da 1 a 255 lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera
  - Non può terminare con un trattino o contenere due trattini consecutivi

  > **Note**
  >
  > Questo valore è archiviato come stringa in caratteri minuscoli.
- Description (nella CLI: `--description`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La descrizione del gruppo di parametri database.
- Tags (nella CLI: `--tags`): un array di oggetti Tag.

  I tag da assegnare al nuovo gruppo di parametri database.

Risposta

Contiene i dettagli di un gruppo di parametri database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called “DescribeDBParameterGroups”.

- DBParameterGroupArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per il gruppo di parametri database.
- DBParameterGroupFamily: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome della famiglia del gruppo di parametri database con la quale è compatibile questo gruppo di parametri database.
- DBParameterGroupName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del gruppo di parametri database.
- Description: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce la descrizione specificata dall'utente per questo gruppo di parametri database.

Errori

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

CreateDBClusterParameterGroup (operazione)

Il nome AWS CLI per questa API è: `create-db-cluster-parameter-group`.

Crea un nuovo gruppo di parametri del cluster di database.

I parametri in un gruppo di parametri del cluster di database si applicano a tutte le istanze in un cluster di database.

Un gruppo di parametri del cluster di database viene creato inizialmente con i parametri predefiniti per il motore di database utilizzato dalle istanze nel cluster di database. Per offrire i valori personalizzati per uno qualsiasi dei parametri, è necessario modificare il gruppo dopo averlo creato utilizzando the section called “ModifyDBClusterParameterGroup”. Dopo aver creato un gruppo di parametri del cluster di database, è necessario associarlo a un cluster di database utilizzando the section called “ModifyDBCluster”. Quando si associa un nuovo gruppo di parametri del cluster di database a un cluster di database in esecuzione, è necessario riavviare le istanze database nel cluster di database

senza failover affinché il nuovo gruppo di parametri del cluster di database e le relative impostazioni diventino effettivi.

> **Important**
>
> Dopo aver creato un gruppo di parametri del cluster database, è necessario attendere almeno 5 minuti prima di creare il primo cluster database che utilizza il gruppo di parametri del cluster database come gruppo predefinito. In questo modo, Amazon Neptune può completare la creazione dell'operazione prima che il gruppo di parametri del cluster di database venga utilizzato come predefinito per un nuovo cluster di database. Questo è particolarmente importante per parametri che sono critici durante la creazione del database predefinito per un cluster database, ad esempio il set di caratteri per il database predefinito specificato dal parametro `character_set_database`. È possibile utilizzare l'opzione Gruppi di parametri della console Amazon Neptune o il comando the section called “DescribeDBClusterParameters” per verificare che il gruppo di parametri del cluster di database sia stato creato o modificato.

Richiesta

- DBClusterParameterGroupName (nella CLI: `--db-cluster-parameter-group-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di parametri del cluster database.

  Vincoli:

  - Deve corrispondere al nome di un oggetto DBClusterParameterGroup esistente.

  > **Note**
  >
  > Questo valore è archiviato come stringa in caratteri minuscoli.

- DBParameterGroupFamily (nella CLI: `--db-parameter-group-family`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della famiglia del gruppo di parametri del cluster di database. Un gruppo di parametri del cluster di database può essere associato a una sola famiglia di gruppo di parametri del cluster di database e può essere applicata solo a un cluster di database per l'esecuzione di un motore di

database e versione del motore compatibile con tale famiglia di gruppo di parametri del cluster di database.

- Description (nella CLI: `--description`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La descrizione del gruppo di parametri del cluster di database.

- Tags (nella CLI: `--tags`): un array di oggetti Tag.

  I tag da assegnare al nuovo gruppo di parametri del cluster di database.

Risposta

Contiene i dettagli di un gruppo di parametri del cluster database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called "DescribeDBClusterParameterGroups".

- DBClusterParameterGroupArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per il gruppo di parametri del cluster database.

- DBClusterParameterGroupName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del gruppo di parametri del cluster database.

- DBParameterGroupFamily: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome della famiglia del gruppo di parametri database con cui è compatibile questo gruppo di parametri del cluster database.

- Description: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce la descrizione specificata dall'utente per questo gruppo di parametri del cluster database.

Errori

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

DeleteDBParameterGroup (operazione)

Il nome AWS CLI per questa API è: `delete-db-parameter-group`.

Elimina uno specifico DBParameterGroup. Il DBParameterGroup da eliminare non può essere associato ad alcune istanze database.

Richiesta

- DBParameterGroupName (nella CLI: `--db-parameter-group-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di parametri database.

  Vincoli:
  - Deve essere il nome di un gruppo di parametri database
  - Non è possibile eliminare un gruppo di parametri database predefinito.
  - Non può essere associato a qualsiasi istanza database

Risposta

- Nessun parametro di risposta.

Errori

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

DeleteDBClusterParameterGroup (operazione)

Il nome AWS CLI per questa API è: `delete-db-cluster-parameter-group`.

Elimina un gruppo di parametri del cluster di database specificato. Il gruppo di parametri del cluster di database da eliminare non può essere associato a qualsiasi cluster di database.

Richiesta

- DBClusterParameterGroupName (nella CLI: `--db-cluster-parameter-group-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

Il nome del gruppo di parametri del cluster database.

Vincoli:

- Deve essere il nome di un gruppo di parametri del cluster di database esistente.
- Non è possibile eliminare un gruppo di parametri del cluster di database predefinito.
- Non può essere associato a qualsiasi cluster di database.

Risposta

- Nessun parametro di risposta.

Errori

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

ModifyDBParameterGroup (operazione)

Il nome AWS CLI per questa API è: `modify-db-parameter-group`.

Modifica i parametri di un gruppo di parametri database. Per modificare più di un parametro, inviare un elenco di quanto segue: `ParameterName`, `ParameterValue` e `ApplyMethod`. Un massimo di 20 parametri possono essere modificati in una singola richiesta.

> **Note**
>
> Le modifiche apportate ai parametri dinamici vengono applicate immediatamente. Affinché le modifiche apportate ai parametri statici diventino effettive, è necessario un riavvio senza failover dell'istanza database associata al gruppo di parametri.

> **Important**
>
> Dopo aver modificato un gruppo di parametri database, è necessario attendere almeno 5 minuti prima di creare la prima istanza database che utilizza il gruppo di parametri database come gruppo predefinito. In questo modo, Amazon Neptune può completare la modifica

dell'operazione prima che il gruppo di parametri venga utilizzato come predefinito per una nuova istanza database. Questo è particolarmente importante per parametri critici durante la creazione del database predefinito per un'istanza database, ad esempio il set di caratteri per il database predefinito specificato dal parametro `character_set_database`. È possibile utilizzare l'opzione Gruppi di parametri della console Amazon Neptune o il comando DescribeDBParameters per verificare che il gruppo di parametri database sia stato creato o modificato.

Richiesta

- DBParameterGroupName (nella CLI: `--db-parameter-group-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di parametri database.

  Vincoli:

  - Se viene specificato, deve corrispondere al nome di un oggetto DBParameterGroup esistente.
- Parameters (nella CLI: `--parameters`): Obbligatorio: un array di oggetti Parametro.

  Un array di nomi di parametri, valori e metodo di applicazione per l'aggiornamento del parametro. È necessario fornire almeno un nome di parametro, un valore e un metodo di applicazione; gli argomenti successivi sono facoltativi. Un massimo di 20 parametri possono essere modificati in una singola richiesta.

  Valori validi (per il metodo di applicazione): `immediate | pending-reboot`

  > **Note**
  >
  > È possibile utilizzare il valore immediato solo con i parametri dinamici. È possibile utilizzare il valore riavvio in attesa per parametri dinamici e statici e le modifiche vengono applicate quando si riavvia l'istanza database senza failover.

Risposta

- DBParameterGroupName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del gruppo di parametri database.

Errori

- DBParameterGroupNotFoundFault
- InvalidDBParameterGroupStateFault

ModifyDBClusterParameterGroup (operazione)

Il nome AWS CLI per questa API è: `modify-db-cluster-parameter-group`.

Modifica i parametri di un gruppo di parametri del cluster di database. Per modificare più di un parametro, inviare un elenco di quanto segue: `ParameterName`, `ParameterValue` e `ApplyMethod`. Un massimo di 20 parametri possono essere modificati in una singola richiesta.

> **Note**
>
> Le modifiche apportate ai parametri dinamici vengono applicate immediatamente. Affinché le modifiche apportate ai parametri statici diventino effettive, è necessario un riavvio senza failover del cluster di database associato al gruppo di parametri.

> **Important**
>
> Dopo aver creato un gruppo di parametri del cluster database, è necessario attendere almeno 5 minuti prima di creare il primo cluster database che utilizza il gruppo di parametri del cluster database come gruppo predefinito. In questo modo, Amazon Neptune può completare la creazione dell'operazione prima che il gruppo di parametri venga utilizzato come predefinito per un nuovo cluster di database. Questo è particolarmente importante per parametri che sono critici durante la creazione del database predefinito per un cluster database, ad esempio il set di caratteri per il database predefinito specificato dal parametro `character_set_database`. È possibile utilizzare l'opzione Gruppi di parametri della console Amazon Neptune o il comando the section called “DescribeDBClusterParameters” per verificare che il gruppo di parametri del cluster di database sia stato creato o modificato.

Richiesta

- DBClusterParameterGroupName (nella CLI: `--db-cluster-parameter-group-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

Il nome del gruppo di parametri del cluster di database da modificare.

- Parameters (nella CLI: `--parameters`): Obbligatorio: un array di oggetti Parametro.

  Un elenco di parametri nel gruppo di parametri del cluster di database da modificare.

Risposta

- DBClusterParameterGroupName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di parametri del cluster database.

  Vincoli:

  - Deve contenere da 1 a 255 lettere o numeri.
  - Il primo carattere deve essere una lettera
  - Non può terminare con un trattino o contenere due trattini consecutivi

  > **Note**
  >
  > Questo valore è archiviato come stringa in caratteri minuscoli.

Errori

- DBParameterGroupNotFoundFault
- InvalidDBParameterGroupStateFault

ResetDBParameterGroup (operazione)

Il nome AWS CLI per questa API è: `reset-db-parameter-group`.

Modifica i parametri di un gruppo di parametri database con il valore predefinito del motore/sistema. Per reimpostare i parametri specifici, fornire un elenco di quanto segue: `ParameterName` e `ApplyMethod`. Per reimpostare l'intero gruppo di parametri database, specificare il nome `DBParameterGroup` e i parametri `ResetAllParameters`. Quando si reimposta l'intero gruppo, i parametri dinamici vengono aggiornati immediatamente e i parametri statici vengono impostati su `pending-reboot` affinché diventino effettivi al successivo riavvio dell'istanza database o quando viene eseguita la richiesta `RebootDBInstance`.

Richiesta

- DBParameterGroupName (nella CLI: `--db-parameter-group-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di parametri database.

  Vincoli:

  - Deve corrispondere al nome di un oggetto DBParameterGroup esistente.
- Parameters (nella CLI: `--parameters`): un array di oggetti Parametro.

  Per reimpostare l'intero gruppo di parametri database, specificare il nome `DBParameterGroup` e i parametri `ResetAllParameters`. Per reimpostare i parametri specifici, fornire un elenco di quanto segue: `ParameterName` e `ApplyMethod`. Un massimo di 20 parametri possono essere modificati in una singola richiesta.

  Valori validi (per il metodo di applicazione): `pending-reboot`
- ResetAllParameters (nella CLI: `--reset-all-parameters`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se ripristinare tutti i parametri (`true`) o meno (`false`) nel gruppo di parametri database ai valori predefiniti.

  Impostazione predefinita: `true`

Risposta

- DBParameterGroupName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del gruppo di parametri database.

Errori

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

ResetDBClusterParameterGroup (operazione)

Il nome AWS CLI per questa API è: `reset-db-cluster-parameter-group`.

Modifica i parametri di un gruppo di parametri del cluster di database con il valore predefinito. Per reimpostare i parametri specifici, inviare un elenco di quanto segue: `ParameterName` e `ApplyMethod`. Per reimpostare l'intero gruppo di parametri del cluster di database, specificare i parametri `DBClusterParameterGroupName` e `ResetAllParameters`.

Quando si reimposta l'intero gruppo, i parametri dinamici vengono aggiornati immediatamente e i parametri statici vengono impostati su `pending-reboot` affinché diventino effettivi al successivo riavvio dell'istanza database o quando viene eseguita la richiesta the section called “RebootDBInstance”. È necessario chiamare the section called “RebootDBInstance” per ogni istanza database nel cluster di database al quale si desidera applicare il parametro statico aggiornato.

Richiesta

- DBClusterParameterGroupName (nella CLI: `--db-cluster-parameter-group-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di parametri del cluster di database da reimpostare.
- Parameters (nella CLI: `--parameters`): un array di oggetti Parametro.

  Un elenco di nomi di parametri nel gruppo di parametri del cluster di database da ripristinare ai valori predefiniti. Non è possibile utilizzare questo parametro se il parametro `ResetAllParameters` è impostato su `true`.
- ResetAllParameters (nella CLI: `--reset-all-parameters`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore è impostato su `true` per ripristinare tutti i parametri nel gruppo di parametri del cluster di database ai valori predefiniti, altrimenti è impostato su `false` . Non è possibile utilizzare questo parametro se c'è un elenco di nomi di parametri specificati per il parametro `Parameters`.

Risposta

- DBClusterParameterGroupName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di parametri del cluster database.

Vincoli:

- Deve contenere da 1 a 255 lettere o numeri.
- Il primo carattere deve essere una lettera
- Non può terminare con un trattino o contenere due trattini consecutivi

> **Note**
>
> Questo valore è archiviato come stringa in caratteri minuscoli.

Errori

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

DescribeDBParameters (operazione)

Il nome AWS CLI per questa API è: `describe-db-parameters`.

Restituisce l'elenco di parametri dettagliati per un determinato gruppo di parametri database.

Richiesta

- DBParameterGroupName (nella CLI: `--db-parameter-group-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome di un determinato gruppo di parametri database per il quale restituire dettagli.

  Vincoli:

  - Se viene specificato, deve corrispondere al nome di un oggetto DBParameterGroup esistente.
- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Questo parametro non è attualmente supportato.
- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

Token di paginazione opzionale fornito da una richiesta `DescribeDBParameters` precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.
- Source (nella CLI: `--source`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  I tipi di parametri da restituire.

  Impostazione predefinita: tutti i tipi di parametri restituiti

  Valori validi: `user | system | engine-default`

Risposta

- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.
- Parameters: una matrice di oggetti Parametro.

  Un elenco di valori the section called “Parametro”.

Errori

- DBParameterGroupNotFoundFault

DescribeDBParameterGroups (operazione)

Il nome AWS CLI per questa API è: `describe-db-parameter-groups`.

Restituisce un elenco di descrizioni `DBParameterGroup`. Se viene specificato `DBParameterGroupName`, l'elenco conterrà solo la descrizione del gruppo di parametri database specificato.

Richiesta

- DBParameterGroupName (nella CLI: `--db-parameter-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome di un determinato gruppo di parametri database per il quale restituire dettagli.

  Vincoli:

  - Se viene specificato, deve corrispondere al nome di un oggetto DBClusterParameterGroup esistente.
- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Questo parametro non è attualmente supportato.
- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta `DescribeDBParameterGroups` precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.
- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.

Risposta

- DBParameterGroups: una matrice di oggetti DBParameterGroup.

  Elenco di istanze di the section called “DBParameterGroup”.
- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

Errori

- DBParameterGroupNotFoundFault

DescribeDBClusterParameters (operazione)

Il nome AWS CLI per questa API è: `describe-db-cluster-parameters`.

Restituisce l'elenco di parametri dettagliati per un determinato gruppo di parametri del cluster di database.

Richiesta

- DBClusterParameterGroupName (nella CLI: `--db-cluster-parameter-group-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome di un determinato gruppo di parametri del cluster di database per il quale restituire dettagli di parametro.

  Vincoli:
  - Se viene specificato, deve corrispondere al nome di un oggetto DBClusterParameterGroup esistente.
- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Questo parametro non è attualmente supportato.
- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta `DescribeDBClusterParameters` precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.
- Source (nella CLI: `--source`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un valore che indica la restituzione solo di parametri per una determinata origine. Le origini dei parametri possono essere `engine`, `service` o `customer`.

Risposta

- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta DescribeDBClusterParameters precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.
- Parameters: una matrice di oggetti Parametro.

  Fornisce un elenco di parametri per il gruppo di parametri del cluster di database.

Errori

- DBParameterGroupNotFoundFault

DescribeDBClusterParameterGroups (operazione)

Il nome AWS CLI per questa API è: `describe-db-cluster-parameter-groups`.

Restituisce un elenco di descrizioni `DBClusterParameterGroup`. Se viene specificato un parametro `DBClusterParameterGroupName`, l'elenco contiene solo la descrizione del gruppo di parametri del cluster di database specificato.

Richiesta

- DBClusterParameterGroupName (nella CLI: `--db-cluster-parameter-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome di un determinato gruppo di parametri del cluster di database per il quale restituire dettagli.

  Vincoli:

  - Se viene specificato, deve corrispondere al nome di un oggetto DBClusterParameterGroup esistente.
- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Questo parametro non è attualmente supportato.
- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta `DescribeDBClusterParameterGroups` precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.
- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.

Risposta

- DBClusterParameterGroups: una matrice di oggetti DBClusterParameterGroup.

  Un elenco di gruppi di parametri del cluster di database.
- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta `DescribeDBClusterParameterGroups` precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

Errori

- DBParameterGroupNotFoundFault

DescribeEngineDefaultParameters (operazione)

Il nome AWS CLI per questa API è: `describe-engine-default-parameters`.

Restituisce il motore predefinito e le informazioni del parametro di sistema per il motore di database specificato.

Richiesta

- DBParameterGroupFamily (nella CLI: `--db-parameter-group-family`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della famiglia del gruppo di parametri database.
- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Attualmente non è supportato.
- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta `DescribeEngineDefaultParameters` precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.
- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.

Risposta

Contiene il risultato di una chiamata di successo dell'operazione the section called "DescribeEngineDefaultParameters".

- DBParameterGroupFamily: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome della famiglia del gruppo di parametri database alla quale si applicano i parametri predefiniti del motore.
- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un token di paginazione opzionale fornito da una richiesta EngineDefaults precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.
- Parameters: una matrice di oggetti Parametro.

  Contiene un elenco di parametri predefiniti del motore.

DescribeEngineDefaultClusterParameters (operazione)

Il nome AWS CLI per questa API è: `describe-engine-default-cluster-parameters`.

Restituisce il motore predefinito e le informazioni del parametro di sistema per il motore del cluster di database .

Richiesta

- DBParameterGroupFamily (nella CLI: `--db-parameter-group-family`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome di una famiglia del gruppo di parametri del cluster di database per il quale restituire informazioni di parametro del motore.
- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Questo parametro non è attualmente supportato.
- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta `DescribeEngineDefaultClusterParameters` precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.

Risposta

Contiene il risultato di una chiamata di successo dell'operazione the section called "DescribeEngineDefaultParameters".

- DBParameterGroupFamily: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome della famiglia del gruppo di parametri database alla quale si applicano i parametri predefiniti del motore.
- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un token di paginazione opzionale fornito da una richiesta EngineDefaults precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.
- Parameters: una matrice di oggetti Parametro.

  Contiene un elenco di parametri predefiniti del motore.

Strutture:

Parametro (struttura)

Specifica un parametro.

Campi

- AllowedValues: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo valido di valori per il parametro.

- ApplyMethod: questo è un valore ApplyMethod di tipo `string`(una stringa con codifica UTF-8).

  Indica quando applicare gli aggiornamenti dei parametri.
- ApplyType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il tipo di parametri specifici per il motore.
- DataType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il tipo di dati valido per il parametro.
- Description: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce una descrizione del parametro.
- IsModifiable: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se il parametro può essere modificato (`true`) o meno (`false`). Alcuni parametri presentano implicazioni operative o di sicurezza che evitano la loro modifica.
- MinimumEngineVersion: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  La prima versione del motore sulla quale si può applicare il parametro.
- ParameterName: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del parametro.
- ParameterValue: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il valore del parametro.
- Source: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica l'origine del valore del parametro.

DBParameterGroup (struttura)

Contiene i dettagli di un gruppo di parametri database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called “DescribeDBParameterGroups”.

Campi

- DBParameterGroupArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

L'Amazon Resource Name (ARN) per il gruppo di parametri database.

- DBParameterGroupFamily: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome della famiglia del gruppo di parametri database con la quale è compatibile questo gruppo di parametri database.
- DBParameterGroupName: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del gruppo di parametri database.
- Description: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce la descrizione specificata dall'utente per questo gruppo di parametri database.

`DBParameterGroup` viene utilizzato come elemento di risposta per:

- CopyDBParameterGroup
- CreateDBParameterGroup

DBClusterParameterGroup (struttura)

Contiene i dettagli di un gruppo di parametri del cluster database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called “DescribeDBClusterParameterGroups”.

Campi

- DBClusterParameterGroupArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per il gruppo di parametri del cluster database.
- DBClusterParameterGroupName: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del gruppo di parametri del cluster database.
- DBParameterGroupFamily: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome della famiglia del gruppo di parametri database con cui è compatibile questo gruppo di parametri del cluster database.

- Description: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce la descrizione specificata dall'utente per questo gruppo di parametri del cluster database.

`DBClusterParameterGroup` viene utilizzato come elemento di risposta per:

- CopyDBClusterParameterGroup
- CreateDBClusterParameterGroup

DBParameterGroupStatus (struttura)

Lo stato del gruppo di parametri database.

Questo tipo di dati viene utilizzato come elemento di risposta nelle seguenti operazioni:

- the section called “CreateDBInstance”
- the section called “DeleteDBInstance”
- the section called “ModifyDBInstance”
- the section called “RebootDBInstance”

Campi

- DBParameterGroupName: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di parametri database.
- ParameterApplyStatus: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato degli aggiornamenti del parametro.

API delle sottoreti Neptune

Operazioni:

- CreateDBSubnetGroup (operazione)
- DeleteDBSubnetGroup (operazione)
- ModifyDBSubnetGroup (operazione)

- DescribeDBSubnetGroups (operazione)

Strutture:

- Sottorete (struttura)
- DBSubnetGroup (struttura)

CreateDBSubnetGroup (operazione)

Il nome AWS CLI per questa API è: `create-db-subnet-group`.

Crea un nuovo gruppo di sottoreti di database. I gruppi di sottoreti di database devono contenere almeno una sottorete in almeno due zone di disponibilità nella stessa regione Amazon.

Richiesta

- DBSubnetGroupDescription (nella CLI: `--db-subnet-group-description`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La descrizione per il gruppo di sottoreti DB.
- DBSubnetGroupName (nella CLI: `--db-subnet-group-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di sottoreti DB. Questo valore è archiviato come stringa in caratteri minuscoli.

  Vincoli: devono contenere un massimo di 255 lettere, numeri, punti, trattini bassi, spazi o trattini. Non deve essere predefinito.

  Esempio: `mySubnetgroup`
- SubnetIds (nella CLI: `--subnet-ids`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  ID di sottorete EC2 per il gruppo di sottoreti di database.
- Tags (nella CLI: `--tags`): un array di oggetti Tag.

  I tag da assegnare al nuovo gruppo di sottoreti di database.

Risposta

Contiene i dettagli di un gruppo di sottoreti di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called “DescribeDBSubnetGroups”.

- DBSubnetGroupArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per il gruppo di sottoreti di database.
- DBSubnetGroupDescription: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce la descrizione del gruppo di sottoreti di database.
- DBSubnetGroupName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di sottoreti DB.
- SubnetGroupStatus: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce lo stato del gruppo di sottoreti di database.
- Subnets: una matrice di oggetti Sottorete.

  Include un elenco di elementi the section called “Sottorete”.
- VpcId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il VpcId del gruppo di sottoreti di database.

Errori

- DBSubnetGroupAlreadyExistsFault
- DBSubnetGroupQuotaExceededFault
- DBSubnetQuotaExceededFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

DeleteDBSubnetGroup (operazione)

Il nome AWS CLI per questa API è: `delete-db-subnet-group`.

Eliminare un gruppo di sottoreti di database.

> **Note**
>
> Il gruppo di sottoreti di database specificato non deve essere associato ad alcuna istanza database.

Richiesta

- DBSubnetGroupName (nella CLI: `--db-subnet-group-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di sottoreti di database da eliminare.

  > **Note**
  >
  > Non è possibile eliminare il gruppo di sottoreti di default.

  Vincoli:

  Vincoli: deve corrispondere al nome di un oggetto DBSubnetGroup esistente. Non deve essere predefinito.

  Esempio: `mySubnetgroup`

Risposta

- Nessun parametro di risposta.

Errori

- InvalidDBSubnetGroupStateFault
- InvalidDBSubnetStateFault
- DBSubnetGroupNotFoundFault

ModifyDBSubnetGroup (operazione)

Il nome AWS CLI per questa API è: `modify-db-subnet-group`.

Modifica un gruppo di sottoreti di database esistente. I gruppi di sottoreti di database devono contenere almeno una sottorete in almeno due zone di disponibilità nella stessa regione Amazon.

Richiesta

- DBSubnetGroupDescription (nella CLI: `--db-subnet-group-description`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  La descrizione per il gruppo di sottoreti DB.
- DBSubnetGroupName (nella CLI: `--db-subnet-group-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di sottoreti DB. Questo valore è archiviato come stringa in caratteri minuscoli. Non è possibile modificare il gruppo di sottoreti di default.

  Vincoli: deve corrispondere al nome di un oggetto DBSubnetGroup esistente. Non deve essere predefinito.

  Esempio: `mySubnetgroup`
- SubnetIds (nella CLI: `--subnet-ids`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  ID di sottorete EC2 per il gruppo di sottoreti di database.

Risposta

Contiene i dettagli di un gruppo di sottoreti di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called “DescribeDBSubnetGroups”.

- DBSubnetGroupArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per il gruppo di sottoreti di database.
- DBSubnetGroupDescription: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce la descrizione del gruppo di sottoreti di database.
- DBSubnetGroupName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di sottoreti DB.

- SubnetGroupStatus: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce lo stato del gruppo di sottoreti di database.

- Subnets: una matrice di oggetti Sottorete.

  Include un elenco di elementi the section called “Sottorete”.

- VpcId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il VpcId del gruppo di sottoreti di database.

Errori

- DBSubnetGroupNotFoundFault
- DBSubnetQuotaExceededFault
- SubnetAlreadyInUse
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

DescribeDBSubnetGroups (operazione)

Il nome AWS CLI per questa API è: `describe-db-subnet-groups`.

Restituisce un elenco di descrizioni DBSubnetGroup. Se viene specificato un DBSubnetGroupName, l'elenco conterrà solo la descrizione del DBSubnetGroup specificato.

Per una panoramica degli intervalli CIDR, consulta Tutorial di Wikipedia.

Richiesta

- DBSubnetGroupName (nella CLI: `--db-subnet-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del gruppo di sottoreti di database per il quale restituire i dettagli.

- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Questo parametro non è attualmente supportato.

- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

Token di paginazione opzionale fornito da una richiesta DescribeDBSubnetGroups precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.

Risposta

- DBSubnetGroups: una matrice di oggetti DBSubnetGroup.

  Elenco di istanze di the section called “DBSubnetGroup”.
- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

Errori

- DBSubnetGroupNotFoundFault

Strutture:

Sottorete (struttura)

Specifica una sottorete.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called “DescribeDBSubnetGroups”.

Campi

- SubnetAvailabilityZone: questo è un oggetto AvailabilityZone.

  Specifica la zona di disponibilità EC2 in cui si trova la sottorete.
- SubnetIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'identificatore della sottorete.
- SubnetStatus: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato della sottorete.

DBSubnetGroup (struttura)

Contiene i dettagli di un gruppo di sottoreti di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called "DescribeDBSubnetGroups".

Campi

- DBSubnetGroupArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per il gruppo di sottoreti di database.
- DBSubnetGroupDescription: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce la descrizione del gruppo di sottoreti di database.
- DBSubnetGroupName: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di sottoreti DB.
- SubnetGroupStatus: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce lo stato del gruppo di sottoreti di database.
- Subnets: questo è un array di oggetti Sottorete.

  Include un elenco di elementi the section called "Sottorete".
- VpcId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il VpcId del gruppo di sottoreti di database.

`DBSubnetGroup` viene utilizzato come elemento di risposta per:

- CreateDBSubnetGroup
- ModifyDBSubnetGroup

API di snapshot Neptune

Operazioni:

- CreateDBClusterSnapshot (operazione)
- DeleteDBClusterSnapshot (operazione)
- CopyDBClusterSnapshot (operazione)
- ModifyDbClusterSnapshotAttribute (operazione)
- RestoreDBClusterFromSnapshot (operazione)
- RestoreDBClusterToPointInTime (operazione)
- DescribeDBClusterSnapshots (operazione)
- DescribeDBClusterSnapshotAttributes (operazione)

Strutture:

- DBClusterSnapshot (struttura)
- DBClusterSnapshotAttribute (struttura)
- DBClusterSnapshotAttributesResult (struttura)

CreateDBClusterSnapshot (operazione)

> Il nome AWS CLI per questa API è: `create-db-cluster-snapshot`.

Crea uno snapshot di un cluster di database.

Richiesta

- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

L'identificatore del cluster di database per il quale creare uno snapshot. Questo parametro non fa distinzione tra maiuscole e minuscole.

Vincoli:

- Deve corrispondere all'identificatore di un oggetto DBCluster esistente.

Esempio: `my-cluster1`

- DBClusterSnapshotIdentifier (nella CLI: `--db-cluster-snapshot-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore dello snapshot del cluster di database. Questo parametro è archiviato come stringa in minuscolo.

  Vincoli:

  - Deve contenere da 1 a 63 lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.

  Esempio: `my-cluster1-snapshot1`

- Tags (nella CLI: `--tags`): un array di oggetti Tag.

  I tag da assegnare allo snapshot del cluster di database.

Risposta

Contiene i dettagli per uno snapshot del cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called "DescribeDBClusterSnapshots".

- AllocatedStorage: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica le dimensioni dello storage allocato, in gibibyte (GiB).

- AvailabilityZones: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 nelle quali è possibile ripristinare istanze nello snapshot del cluster di database.

- ClusterCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).

- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'identificatore del cluster di database, dal quale è stato creato questo snapshot del cluster di database.

- DBClusterSnapshotArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per lo snapshot del cluster di database.

- DBClusterSnapshotIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'identificatore per uno snapshot del cluster DB. Deve corrispondere all'identificatore di una snapshot esistente.

  Dopo aver ripristinato un cluster di database utilizzando `DBClusterSnapshotIdentifier`, è necessario specificare lo stesso `DBClusterSnapshotIdentifier` per eventuali aggiornamenti futuri al cluster di database. Quando specifichi questa proprietà per un aggiornamento, il cluster di database non viene nuovamente ripristinato dallo snapshot e i dati nel database non vengono modificati.

  Tuttavia, se non specifichi `DBClusterSnapshotIdentifier`, viene creato un cluster DB vuoto e il cluster DB originale viene eliminato. Se specifichi una proprietà diversa dalla proprietà di ripristino dello snapshot precedente, il cluster DB viene ripristinato dallo snapshot specificato da `DBClusterSnapshotIdentifier` e il cluster DB originale viene eliminato.

- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del motore di database.

- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce la versione del motore di database da utilizzare per questo snapshot del cluster di database.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.

- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per lo snapshot del cluster database crittografato.

- LicenseModel: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce le informazioni del modello di licenza per questo snapshot del cluster di database.

- PercentProgress: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica la percentuale dei dati stimati che sono stati trasferiti.

- Port: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica la porta con la quale il cluster di database ascoltava al momento della creazione dello snapshot.

- SnapshotCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Fornisce l'ora di creazione dello snapshot, nel formato UTC (Universal Coordinated Time).

- SnapshotType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il tipo di snapshot del cluster di database.

- SourceDBClusterSnapshotArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se lo snapshot del cluster di database è stato copiato da uno snapshot del cluster di database di origine, l'Amazon Resource Name (ARN) per lo snapshot del cluster di database di origine, in caso contrario, presenta un valore null.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato dello snapshot del cluster di database.

- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se lo snapshot del cluster di database è crittografato.

- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione associato allo snapshot del cluster di database.

- VpcId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'ID VPC associato allo snapshot del cluster di database.

Errori

- DBClusterSnapshotAlreadyExistsFault
- InvalidDBClusterStateFault
- DBClusterNotFoundFault
- SnapshotQuotaExceededFault
- InvalidDBClusterSnapshotStateFault

DeleteDBClusterSnapshot (operazione)

Il nome AWS CLI per questa API è: `delete-db-cluster-snapshot`.

Elimina uno snapshot del cluster di database. Se si copia lo snapshot, l'operazione di copia viene terminata.

> **Note**
>
> Lo snapshot del cluster di database snapshot deve essere nello stato `available` per essere eliminato.

Richiesta

- DBClusterSnapshotIdentifier (nella CLI: `--db-cluster-snapshot-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore dello snapshot del cluster di database da eliminare.

  Vincoli: deve essere il nome di uno snapshot del cluster di database esistente nello stato `available`.

Risposta

Contiene i dettagli per uno snapshot del cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called "DescribeDBClusterSnapshots".

- AllocatedStorage: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica le dimensioni dello storage allocato, in gibibyte (GiB).

- AvailabilityZones: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 nelle quali è possibile ripristinare istanze nello snapshot del cluster di database.

- ClusterCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).

- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'identificatore del cluster di database, dal quale è stato creato questo snapshot del cluster di database.

- DBClusterSnapshotArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per lo snapshot del cluster di database.

- DBClusterSnapshotIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'identificatore per uno snapshot del cluster DB. Deve corrispondere all'identificatore di una snapshot esistente.

  Dopo aver ripristinato un cluster di database utilizzando `DBClusterSnapshotIdentifier`, è necessario specificare lo stesso `DBClusterSnapshotIdentifier` per eventuali aggiornamenti futuri al cluster di database. Quando specifichi questa proprietà per un aggiornamento, il cluster di database non viene nuovamente ripristinato dallo snapshot e i dati nel database non vengono modificati.

  Tuttavia, se non specifichi `DBClusterSnapshotIdentifier`, viene creato un cluster DB vuoto e il cluster DB originale viene eliminato. Se specifichi una proprietà diversa dalla proprietà di ripristino dello snapshot precedente, il cluster DB viene ripristinato dallo snapshot specificato da `DBClusterSnapshotIdentifier` e il cluster DB originale viene eliminato.

- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

Specifica il nome del motore di database.

- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce la versione del motore di database da utilizzare per questo snapshot del cluster di database.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.

- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per lo snapshot del cluster database crittografato.

- LicenseModel: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce le informazioni del modello di licenza per questo snapshot del cluster di database.

- PercentProgress: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica la percentuale dei dati stimati che sono stati trasferiti.

- Port: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica la porta con la quale il cluster di database ascoltava al momento della creazione dello snapshot.

- SnapshotCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Fornisce l'ora di creazione dello snapshot, nel formato UTC (Universal Coordinated Time).

- SnapshotType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il tipo di snapshot del cluster di database.

- SourceDBClusterSnapshotArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se lo snapshot del cluster di database è stato copiato da uno snapshot del cluster di database di origine, l'Amazon Resource Name (ARN) per lo snapshot del cluster di database di origine, in caso contrario, presenta un valore null.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

Specifica lo stato dello snapshot del cluster di database.

- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se lo snapshot del cluster di database è crittografato.

- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione associato allo snapshot del cluster di database.

- VpcId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'ID VPC associato allo snapshot del cluster di database.

Errori

- InvalidDBClusterSnapshotStateFault
- DBClusterSnapshotNotFoundFault

CopyDBClusterSnapshot (operazione)

Il nome AWS CLI per questa API è: `copy-db-cluster-snapshot`.

Copia uno snapshot di un cluster di database.

Per copiare uno snapshot del cluster di database da uno snapshot del cluster di database manuale condiviso, `SourceDBClusterSnapshotIdentifier` deve essere l'Amazon Resource Name (ARN) dello snapshot del cluster di database condiviso.

Richiesta

- CopyTags (nella CLI: `--copy-tags`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  True per copiare tutti i tag dallo snapshot del cluster di database di origine allo snapshot del cluster di database di destinazione, in caso contrario false. Il valore di default è false.

- KmsKeyId (nella CLI: `--kms-key-id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

L'ID della chiave Amazon KMS per uno snapshot del cluster database crittografato. L'ID della chiave KMS è l'Amazon Resource Name (ARN), l'identificatore della chiave KMS o l'alias della chiave KMS per la chiave di crittografia KMS.

Se copi uno snapshot del cluster database crittografato dall'account Amazon, puoi specificare un valore per `KmsKeyId` per crittografare la copia con una nuova chiave di crittografia KMS. Se non si specifica un valore per `KmsKeyId`, la copia dello snapshot del cluster di database viene crittografata con la stessa chiave KMS dello snapshot del cluster di database di origine.

Se copi uno snapshot del cluster database crittografato che è condiviso da un altro account Amazon, devi specificare un valore per `KmsKeyId`.

Le chiavi di crittografia KMS sono specifiche per la regione Amazon nella quale vengono create e non puoi utilizzare le chiavi di crittografia di una regione Amazon in un'altra.

Non è possibile crittografare uno snapshot del cluster di database non crittografato durante la sua copia. Se si tenta di copiare uno snapshot del cluster di database non crittografato e si specifica un valore per il parametro KmsKeyId, viene restituito un errore.

- PreSignedUrl (nella CLI: `--pre-signed-url`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Attualmente non è supportato.

- SourceDBClusterSnapshotIdentifier (nella CLI: `--source-db-cluster-snapshot-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore dello snapshot del cluster di database da copiare. Questo parametro non fa distinzione tra maiuscole e minuscole.

  Vincoli:

  - Deve specificare uno snapshot di sistema valido nello stato "disponibile".
  - Specificare un identificatore dello snapshot di database valido.

  Esempio: `my-cluster-snapshot1`

- Tags (nella CLI: `--tags`): un array di oggetti Tag.

  I tag da assegnare alla nuova copia dello snapshot del cluster di database.

- TargetDBClusterSnapshotIdentifier (nella CLI: `--target-db-cluster-snapshot-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

L'identificatore del nuovo snapshot del cluster di database da creare dallo snapshot del cluster di database di origine. Questo parametro non fa distinzione tra maiuscole e minuscole.

Vincoli:

- Deve contenere da 1 a 63 lettere, numeri o trattini.
- Il primo carattere deve essere una lettera.
- Non può terminare con un trattino o contenere due trattini consecutivi.

Esempio: `my-cluster-snapshot2`

Risposta

Contiene i dettagli per uno snapshot del cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called "DescribeDBClusterSnapshots".

- AllocatedStorage: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica le dimensioni dello storage allocato, in gibibyte (GiB).
- AvailabilityZones: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 nelle quali è possibile ripristinare istanze nello snapshot del cluster di database.
- ClusterCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).
- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'identificatore del cluster di database, dal quale è stato creato questo snapshot del cluster di database.
- DBClusterSnapshotArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per lo snapshot del cluster di database.
- DBClusterSnapshotIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

Specifica l'identificatore per uno snapshot del cluster DB. Deve corrispondere all'identificatore di una snapshot esistente.

Dopo aver ripristinato un cluster di database utilizzando `DBClusterSnapshotIdentifier`, è necessario specificare lo stesso `DBClusterSnapshotIdentifier` per eventuali aggiornamenti futuri al cluster di database. Quando specifichi questa proprietà per un aggiornamento, il cluster di database non viene nuovamente ripristinato dallo snapshot e i dati nel database non vengono modificati.

Tuttavia, se non specifichi `DBClusterSnapshotIdentifier`, viene creato un cluster DB vuoto e il cluster DB originale viene eliminato. Se specifichi una proprietà diversa dalla proprietà di ripristino dello snapshot precedente, il cluster DB viene ripristinato dallo snapshot specificato da `DBClusterSnapshotIdentifier` e il cluster DB originale viene eliminato.

- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del motore di database.

- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce la versione del motore di database da utilizzare per questo snapshot del cluster di database.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.

- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per lo snapshot del cluster database crittografato.

- LicenseModel: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce le informazioni del modello di licenza per questo snapshot del cluster di database.

- PercentProgress: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Specifica la percentuale dei dati stimati che sono stati trasferiti.

- Port: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

Specifica la porta con la quale il cluster di database ascoltava al momento della creazione dello snapshot.

- SnapshotCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Fornisce l'ora di creazione dello snapshot, nel formato UTC (Universal Coordinated Time).

- SnapshotType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il tipo di snapshot del cluster di database.

- SourceDBClusterSnapshotArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se lo snapshot del cluster di database è stato copiato da uno snapshot del cluster di database di origine, l'Amazon Resource Name (ARN) per lo snapshot del cluster di database di origine, in caso contrario, presenta un valore null.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato dello snapshot del cluster di database.

- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se lo snapshot del cluster di database è crittografato.

- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione associato allo snapshot del cluster di database.

- VpcId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'ID VPC associato allo snapshot del cluster di database.

Errori

- DBClusterSnapshotAlreadyExistsFault
- DBClusterSnapshotNotFoundFault
- InvalidDBClusterStateFault
- InvalidDBClusterSnapshotStateFault
- SnapshotQuotaExceededFault
- KMSKeyNotAccessibleFault

ModifyDbClusterSnapshotAttribute (operazione)

Il nome AWS CLI per questa API è: `modify-db-cluster-snapshot-attribute`.

Aggiunge un attributo e valori a, o rimuove un attributo e valori da, uno snapshot del cluster di database manuale.

Per condividere uno snapshot del cluster database manuale con altri account Amazon, specifica `restore` come `AttributeName` e utilizza il parametro `ValuesToAdd` per aggiungere un elenco di ID degli account Amazon che sono autorizzati a ripristinare lo snapshot del cluster database manuale. Utilizza il valore `all` per rendere pubblico lo snapshot del cluster database manuale, il che significa che può essere copiato o ripristinato da tutti gli account Amazon. Non aggiungere il valore `all` per nessuno snapshot del cluster database manuale contenente informazioni private che non vuoi rendere disponibili a tutti gli account Amazon. Se uno snapshot del cluster database manuale è crittografato, può essere condiviso, ma solo specificando un elenco di ID account Amazon autorizzati per il parametro `ValuesToAdd`. Non è possibile utilizzare `all` come valore per il parametro in questo caso.

Per visualizzare quali account Amazon hanno accesso per copiare o ripristinare uno snapshot del cluster database manuale oppure se uno snapshot del cluster database manuale è pubblico o privato, utilizza l'azione API the section called “DescribeDBClusterSnapshotAttributes”.

Richiesta

- AttributeName (nella CLI: `--attribute-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome dell'attributo dello snapshot del cluster di database da modificare.

  Per gestire le autorizzazioni per altri account Amazon per copiare o ripristinare uno snapshot del cluster database manuale, imposta questo valore su `restore`.
- DBClusterSnapshotIdentifier (nella CLI: `--db-cluster-snapshot-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore per lo snapshot del cluster di database per il quale modificare gli attributi.
- ValuesToAdd (nella CLI: `--values-to-add`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di attributi dello snapshot del cluster di database da aggiungere all'attributo specificato da `AttributeName`.

Per autorizzare altri account Amazon a copiare o ripristinare uno snapshot del cluster database manuale, imposta questo elenco affinché includa uno o più ID account Amazon o `all` per rendere ripristinabile lo snapshot del cluster database manuale da qualsiasi account Amazon. Non aggiungere il valore `all` per nessuno snapshot del cluster database manuale contenente informazioni private che non vuoi rendere disponibili a tutti gli account Amazon.

- ValuesToRemove (nella CLI: `--values-to-remove`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di attributi dello snapshot del cluster di database da rimuovere dall'attributo specificato da `AttributeName`.

  Per rimuovere l'autorizzazione per altri account Amazon a copiare o ripristinare uno snapshot del cluster database manuale, imposta questo elenco per includere uno o più identificatori di account Amazon o `all` per rimuovere l'autorizzazione di qualsiasi account Amazon a copiare o ripristinare lo snapshot del cluster database. Se specifichi `all`, un account Amazon il cui ID account viene esplicitamente aggiunto all'attributo `restore` può continuare a copiare o ripristinare uno snapshot del cluster database manuale.

Risposta

Contiene i risultati di una chiamata riuscita all'operazione API the section called “DescribeDBClusterSnapshotAttributes”.

Gli attributi dello snapshot del cluster database manuale vengono utilizzati per autorizzare altri account Amazon a copiare o ripristinare uno snapshot del cluster database manuale. Per ulteriori informazioni, consultare the section called “ModifyDBClusterSnapshotAttribute” operazione API.

- DBClusterSnapshotAttributes: una matrice di oggetti DBClusterSnapshotAttribute.

  L'elenco di attributi e valori per lo snapshot del cluster di database manuale.

- DBClusterSnapshotIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore dello snapshot del cluster di database manuale al quale si applicano gli attributi.

Errori

- DBClusterSnapshotNotFoundFault

- InvalidDBClusterSnapshotStateFault
- SharedSnapshotQuotaExceededFault

RestoreDBClusterFromSnapshot (operazione)

Il nome AWS CLI per questa API è: `restore-db-cluster-from-snapshot`.

Crea un nuovo cluster database da uno snapshot di database o da uno snapshot del cluster database.

Se uno snapshot DB viene specificato, il cluster di database di destinazione viene creato dallo snapshot di database di origine con una configurazione predefinita e il gruppo di sicurezza predefiniti.

Se viene specificato uno snapshot del cluster di database, il cluster di database di destinazione viene creato dal punto di ripristino del cluster di database di origine con la stessa configurazione del cluster originale di database di origine, però il nuovo cluster di database viene creato con il gruppo di sicurezza predefinito.

Richiesta

- AvailabilityZones (nella CLI: `--availability-zones`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 nelle quali è possibile creare istanze nel cluster di database ripristinato.
- CopyTagsToSnapshot (nella CLI: `--copy-tags-to-snapshot`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su *`true`*, i tag vengono copiati in qualsiasi snapshot del cluster database ripristinato che viene creato.
- DatabaseName (nella CLI: `--database-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato.
- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del cluster di database da creare dallo snapshot di database o dallo snapshot del cluster di database. Questo parametro non fa distinzione tra maiuscole e minuscole.

Vincoli:

- Deve contenere da 1 a 63 lettere, numeri o trattini
- Il primo carattere deve essere una lettera
- Non può terminare con un trattino o contenere due trattini consecutivi

Esempio: `my-snapshot-id`

- DBClusterParameterGroupName (nella CLI: `--db-cluster-parameter-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del gruppo di parametri del cluster di database che desideri associare al nuovo cluster di database.

  Vincoli:

  - Se viene specificato, deve corrispondere al nome di un oggetto DBClusterParameterGroup esistente.
- DBSubnetGroupName (nella CLI: `--db-subnet-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del gruppo di sottoreti di database da utilizzare per il nuovo cluster di database.

  Vincoli: se specificato, deve corrispondere al nome di un oggetto DBSubnetGroup esistente.

  Esempio: `mySubnetgroup`
- DeletionProtection (nella CLI: `--deletion-protection`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se per il cluster di database è abilitata la protezione dall'eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione. Per impostazione predefinita, la protezione da eliminazione è disabilitata.
- EnableCloudwatchLogsExports (nella CLI: `--enable-cloudwatch-logs-exports`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'elenco dei log che il cluster di database ripristinato deve esportare in Amazon CloudWatch Logs.
- EnableIAMDatabaseAuthentication (nella CLI: `--enable-iam-database-authentication`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

True per abilitare la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database, in caso contrario false.

Impostazione predefinita: `false`

- Engine (nella CLI: `--engine`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il motore di database da utilizzare per il nuovo cluster di database.

  Impostazione predefinita: uguale all'origine

  Vincolo: deve essere compatibile con il motore dell'origine

- EngineVersion (nella CLI: `--engine-version`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  La versione del motore di database da utilizzare per il nuovo cluster di database.

- KmsKeyId (nella CLI: `--kms-key-id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore della chiave Amazon KMS da utilizzare per il ripristino di un cluster database crittografato da uno snapshot di database o da uno snapshot del cluster database.

  L'identificatore della chiave KMS è l'Amazon Resource Name (ARN) per la chiave di crittografia KMS. Se stai ripristinando un cluster database con lo stesso account Amazon che possiede la chiave di crittografia KMS utilizzata per crittografare il nuovo cluster database, puoi utilizzare l'alias della chiave KMS al posto dell'ARN per la chiave di crittografia KMS.

  Se non si specifica un valore per il parametro `KmsKeyId`, avviene quanto segue:

  - Se lo snapshot di database o lo snapshot del cluster database in `SnapshotIdentifier` sono crittografati, il cluster database viene crittografato utilizzando la chiave KMS utilizzata per crittografare lo snapshot di database o lo snapshot del cluster database.
  - Se lo snapshot di database o lo snapshot del cluster database in `SnapshotIdentifier` non sono crittografati, il cluster database ripristinato non è crittografato.

- Port (nella CLI: `--port`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Il numero della porta sulla quale il nuovo cluster di database accetta connessioni.

Vincoli: il valore deve essere compreso nell'intervallo `1150-65535`

Impostazione predefinita: la stessa porta del cluster di database originale.

- ServerlessV2ScalingConfiguration (nella CLI: `--serverless-v2-scaling-configuration`): un oggetto ServerlessV2ScalingConfiguration.

  Contiene la configurazione di dimensionamento di un cluster database Neptune Serverless.

  Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

- SnapshotIdentifier (nella CLI: `--snapshot-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore per lo snapshot di database o dello snapshot del cluster di database dal quale ripristinare.

  È possibile utilizzare il nome o l'Amazon Resource Name (ARN) per specificare uno snapshot del cluster di database. Tuttavia, è possibile utilizzare solo l'ARN per specificare uno snapshot di database.

  Vincoli:

  - Deve corrispondere all'identificatore di uno snapshot esistente.

- StorageType (nella CLI: `--storage-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il tipo di archiviazione da associare al cluster di database.

  Valori validi: `standard`, `iopt1`

  Impostazione predefinita: `standard`

- Tags (nella CLI: `--tags`): un array di oggetti Tag.

  I tag da assegnare al cluster di database ripristinato.

- VpcSecurityGroupIds (nella CLI: `--vpc-security-group-ids`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco dei gruppi di sicurezza VPC al quale appartiene il nuovo cluster di database.

Risposta

Contiene i dettagli di un cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'the section called “DescribeDBClusters”.

- AllocatedStorage: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  `AllocatedStorage` restituisce sempre 1, perché le dimensioni di storage dei cluster di database Neptune non sono fisse, ma vengono regolate automaticamente in base alle esigenze.
- AssociatedRoles: una matrice di oggetti DBClusterRole.

  Fornisce un elenco dei ruoli Amazon Identity and Access Management (IAM) associati al cluster di database. I ruoli IAM associati a un cluster di database concedono l'autorizzazione per l'accesso del cluster di database ad altri servizi Amazon per tuo conto.
- AutomaticRestartTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Ora in cui il cluster database verrà riavviato automaticamente.
- AvailabilityZones: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 in cui è possibile creare istanze nel cluster di database.
- BacktrackConsumedChangeRecords: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.
- BacktrackWindow: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.
- BackupRetentionPeriod: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.
- Capacity: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Non supportato da Neptune.

- CloneGroupId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identifica il gruppo di cloni a cui è associato il cluster di database.

- ClusterCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).

- CopyTagsToSnapshot: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su *`true`*, i tag vengono copiati in qualsiasi snapshot del cluster database che viene creato.

- CrossAccountClone: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `true`, il cluster database può essere clonato su più account.

- DatabaseName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nome del database iniziale di questo cluster di database, fornito al momento della creazione, se è stato specificato un nome quando il cluster di database è stato creato. Questo stesso nome viene restituito per la durata del cluster di database.

- DBClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Amazon Resource Name (ARN) per il cluster di database.

- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore del cluster di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un cluster di database.

- DBClusterMembers: una matrice di oggetti DBClusterMember.

  Fornisce l'elenco delle istanze che compongono il cluster di database.

- DBClusterParameterGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del gruppo di parametri del cluster di database per il cluster stesso.

- DbClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

Identificatore univoco e non modificabile della regione Amazon per il cluster database. Questo identificativo è disponibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per il cluster database.

- DBSubnetGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica le informazioni sul gruppo di sottoreti associato al cluster di database, tra cui nome, descrizione e sottoreti nel gruppo.
- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se per il cluster di database è abilitata o meno la protezione da eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione.
- EarliestBacktrackTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Non supportato da Neptune.
- EarliestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica il primo orario possibile per il ripristino point-in-time di un database.
- EnabledCloudwatchLogsExports: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questo cluster database può esportare in CloudWatch Logs. I tipi di log validi sono: `audit` (per pubblicare log di audit su CloudWatch) e slowquery (per pubblicare log di slow query su CloudWatch). Consulta Pubblicazione di log Neptune nei file di log Amazon CloudWatch.
- Endpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'endpoint di connessione per l'istanza primaria del cluster di database.
- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questo cluster di database.
- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.
- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.

- HostedZoneId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'ID che Amazon Route 53 assegna al momento della creazione di una zona ospitata.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.

- IOOptimizedNextAllowedModificationTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La prossima volta puoi modificare il cluster di database per utilizzare il tipo di archiviazione `iopt1`.

- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per il cluster di database crittografato.

- LatestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.

- MultiAZ: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database ha istanze in più zone di disponibilità.

- PendingModifiedValues: un oggetto ClusterPendingModifiedValues.

  Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione `ModifyDBCluster` e contiene le modifiche che verranno applicate nella finestra di manutenzione successiva.

- PercentProgress: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'avanzamento dell'operazione sotto forma di percentuale.

- Port: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto il motore di database.

- PreferredBackupWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.

- ReaderEndpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'endpoint di lettura per il cluster di database. L'endpoint di lettura per un cluster di database bilancia il carico delle connessioni tra le repliche di lettura disponibili in un cluster di database. Man mano che i client richiedono nuove connessioni all'endpoint di lettura, Neptune distribuisce tali richieste fra le repliche di lettura nel cluster di database. Questa funzionalità è utile per bilanciare il carico di lavoro di lettura tra più repliche di lettura nel cluster di database.

  Se si verifica un failover e la replica di lettura a cui sei connesso viene promossa a istanza primaria, la connessione viene interrotta. Per continuare a inviare il carico di lavoro di lettura ad altre repliche di lettura nel cluster, puoi quindi riconnetterti all'endpoint di lettura.

- ReadReplicaIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori delle repliche di lettura associate a questo cluster di database.

- ReplicationSourceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ReplicationType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ServerlessV2ScalingConfiguration: un oggetto ServerlessV2ScalingConfigurationInfo.

  Mostra la configurazione del dimensionamento per un cluster database Neptune Serverless.

  Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del cluster di database.

- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

Specifica se il cluster di database è crittografato.

- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione utilizzato dal cluster di database.

  Valori validi:

  - **`standard`**: (impostazione predefinita) fornisce archiviazione di database dai costi contenuti per applicazioni con un numero di operazioni di I/O da moderato a basso.
  - **`iopt1`**: abilita l'archiviazione ottimizzata per l'I/O che è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O che richiedono prezzi prevedibili con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

    L'archiviazione ottimizzata per l'I/O di Neptune è disponibile solo a partire dal rilascio 1.3.0.0 del motore.
- VpcSecurityGroups: una matrice di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco dei gruppi di sicurezza VPC a cui appartiene il cluster di database.

Errori

- DBClusterAlreadyExistsFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterSnapshotNotFoundFault
- InsufficientDBClusterCapacityFault
- InsufficientStorageClusterCapacityFault
- InvalidDBSnapshotStateFault
- InvalidDBClusterSnapshotStateFault
- StorageQuotaExceededFault
- InvalidVPCNetworkStateFault
- InvalidRestoreFault

- DBSubnetGroupNotFoundFault
- InvalidSubnet
- OptionGroupNotFoundFault
- KMSKeyNotAccessibleFault
- DBClusterParameterGroupNotFoundFault

RestoreDBClusterToPointInTime (operazione)

Il nome AWS CLI per questa API è: `restore-db-cluster-to-point-in-time`.

Ripristina un cluster di database a un point-in-time arbitrario. Gli utenti possono ripristinare a qualsiasi point-in-time prima di `LatestRestorableTime` per un massimo di `BackupRetentionPeriod` giorni. Il cluster di database di destinazione viene creato dal punto di ripristino del cluster di database di origine con la stessa configurazione del cluster originale di database di origine, però il nuovo cluster di database viene creato con il gruppo di sicurezza DB predefinito.

> **Note**
>
> Questa operazione ripristina solo il cluster di database, non le istanze database per tale cluster di database. È necessario invocare l'operazione the section called “CreateDBInstance” per creare le istanze database del cluster di database ripristinato, specificando l'identificatore del cluster di database ripristinato in `DBClusterIdentifier`. È possibile creare le istanze database solo dopo che l'operazione `RestoreDBClusterToPointInTime` è completata e il cluster di database è disponibile.

Richiesta

- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del nuovo cluster di database da creare.

  Vincoli:
  - Deve contenere da 1 a 63 lettere, numeri o trattini
  - Il primo carattere deve essere una lettera
  - Non può terminare con un trattino o contenere due trattini consecutivi

- DBClusterParameterGroupName (nella CLI: `--db-cluster-parameter-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del gruppo di parametri del cluster di database che desideri associare al nuovo cluster di database.

  Vincoli:

  - Se viene specificato, deve corrispondere al nome di un oggetto DBClusterParameterGroup esistente.
- DBSubnetGroupName (nella CLI: `--db-subnet-group-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del gruppo di sottoreti di database da utilizzare per il nuovo cluster di database.

  Vincoli: se specificato, deve corrispondere al nome di un oggetto DBSubnetGroup esistente.

  Esempio: `mySubnetgroup`
- DeletionProtection (nella CLI: `--deletion-protection`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se per il cluster di database è abilitata la protezione dall'eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione. Per impostazione predefinita, la protezione da eliminazione è disabilitata.
- EnableCloudwatchLogsExports (nella CLI: `--enable-cloudwatch-logs-exports`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'elenco dei log che il cluster di database ripristinato deve esportare in CloudWatch Logs.
- EnableIAMDatabaseAuthentication (nella CLI: `--enable-iam-database-authentication`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  True per abilitare la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database, in caso contrario false.

  Impostazione predefinita: `false`
- KmsKeyId (nella CLI: `--kms-key-id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore della chiave Amazon KMS da utilizzare per il ripristino di un cluster database crittografato da un cluster database crittografato.

L'identificatore della chiave KMS è l'Amazon Resource Name (ARN) per la chiave di crittografia KMS. Se stai ripristinando un cluster database con lo stesso account Amazon che possiede la chiave di crittografia KMS utilizzata per crittografare il nuovo cluster database, puoi utilizzare l'alias della chiave KMS al posto dell'ARN per la chiave di crittografia KMS.

È possibile ripristinare a un nuovo cluster di database e crittografare il nuovo cluster di database con una chiave KMS diversa dalla chiave KMS utilizzata per crittografare il cluster di database di origine. Il nuovo cluster di database è crittografato con la chiave KMS identificata dal parametro `KmsKeyId`.

Se non si specifica un valore per il parametro `KmsKeyId`, avviene quanto segue:

- Se il cluster di database è crittografato, il cluster di database ripristinato viene crittografato utilizzando la chiave KMS utilizzata per crittografare il cluster di database di origine.
- Se il cluster di database non è crittografato, il cluster di database ripristinato non è crittografato.

Se `DBClusterIdentifier` si riferisce a un cluster di database che non è crittografato, la richiesta di ripristino viene respinta.

- Port (nella CLI: `--port`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Il numero della porta sulla quale il nuovo cluster di database accetta connessioni.

  Vincoli: il valore deve essere compreso nell'intervallo `1150-65535`

  Impostazione predefinita: la stessa porta del cluster di database originale.
- RestoreToTime (nella CLI: `--restore-to-time`): un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La data e l'ora alle quali ripristinare il cluster di database.

  Valori validi: il valore deve essere espresso in formato Universal Coordinated Time (UTC)

  Vincoli:

  - Deve essere prima dell'ultima ora di ripristino per l'istanza database
  - Deve essere specificato se il parametro `UseLatestRestorableTime` non viene fornito
  - Non può essere specificato se il parametro `UseLatestRestorableTime` è true
  - Non può essere specificato se il parametro `RestoreType` è `copy-on-write`

Esempio: `2015-03-07T23:45:00Z`

- RestoreType (nella CLI: `--restore-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di ripristino da eseguire. È possibile specificare uno dei seguenti valori:

  - `full-copy`: il nuovo cluster database viene ripristinato come una copia completa del cluster database di origine.
  - `copy-on-write`: il nuovo cluster database viene ripristinato come un clone del cluster database di origine.

  Se non si specifica un valore `RestoreType`, il nuovo cluster di database viene ripristinato come una copia completa del cluster di database di origine.

- ServerlessV2ScalingConfiguration (nella CLI: `--serverless-v2-scaling-configuration`): un oggetto ServerlessV2ScalingConfiguration.

  Contiene la configurazione di dimensionamento di un cluster database Neptune Serverless.

  Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

- SourceDBClusterIdentifier (nella CLI: `--source-db-cluster-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore del cluster di database di origine dal quale eseguire il ripristino.

  Vincoli:

  - Deve corrispondere all'identificatore di un oggetto DBCluster esistente.

- StorageType (nella CLI: `--storage-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il tipo di archiviazione da associare al cluster di database.

  Valori validi: `standard`, `iopt1`

  Impostazione predefinita: `standard`

- Tags (nella CLI: `--tags`): un array di oggetti Tag.

  I tag da applicare al cluster di database ripristinato.

- UseLatestRestorableTime (nella CLI: `--use-latest-restorable-time`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore impostato su `true` per ripristinare il cluster di database all'ora dell'ultimo backup ripristinabile e altrimenti impostato su `false`.

  Impostazione predefinita: `false`

  Vincoli: non è possibile specificare se il parametro `RestoreToTime` viene fornito.

- VpcSecurityGroupIds (nella CLI: `--vpc-security-group-ids`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco dei gruppi di sicurezza VPC ai quali appartengono i nuovi cluster di database.

Risposta

Contiene i dettagli di un cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'the section called “DescribeDBClusters”.

- AllocatedStorage: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  `AllocatedStorage` restituisce sempre 1, perché le dimensioni di storage dei cluster di database Neptune non sono fisse, ma vengono regolate automaticamente in base alle esigenze.

- AssociatedRoles: una matrice di oggetti DBClusterRole.

  Fornisce un elenco dei ruoli Amazon Identity and Access Management (IAM) associati al cluster di database. I ruoli IAM associati a un cluster di database concedono l'autorizzazione per l'accesso del cluster di database ad altri servizi Amazon per tuo conto.

- AutomaticRestartTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Ora in cui il cluster database verrà riavviato automaticamente.

- AvailabilityZones: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 in cui è possibile creare istanze nel cluster di database.

- BacktrackConsumedChangeRecords: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.

- BacktrackWindow: un valore LongOptional di tipo `long` (numero intero a 64 bit con segno).

  Non supportato da Neptune.

- BackupRetentionPeriod: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica il numero di giorni durante i quali vengono conservati gli snapshot DB automatici.

- Capacity: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Non supportato da Neptune.

- CloneGroupId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identifica il gruppo di cloni a cui è associato il cluster di database.

- ClusterCreateTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).

- CopyTagsToSnapshot: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su *`true`*, i tag vengono copiati in qualsiasi snapshot del cluster database che viene creato.

- CrossAccountClone: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `true`, il cluster database può essere clonato su più account.

- DatabaseName: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene il nome del database iniziale di questo cluster di database, fornito al momento della creazione, se è stato specificato un nome quando il cluster di database è stato creato. Questo stesso nome viene restituito per la durata del cluster di database.

- DBClusterArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

Amazon Resource Name (ARN) per il cluster di database.

- DBClusterIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene un identificatore del cluster di database fornito dall'utente. Questo identificatore è la chiave univoca che identifica un cluster di database.

- DBClusterMembers: una matrice di oggetti DBClusterMember.

  Fornisce l'elenco delle istanze che compongono il cluster di database.

- DBClusterParameterGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del gruppo di parametri del cluster di database per il cluster stesso.

- DbClusterResourceId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore univoco e non modificabile della regione Amazon per il cluster database. Questo identificativo è disponibile nelle voci di log di Amazon CloudTrail ogni volta che si accede alla chiave Amazon KMS per il cluster database.

- DBSubnetGroup: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica le informazioni sul gruppo di sottoreti associato al cluster di database, tra cui nome, descrizione e sottoreti nel gruppo.

- DeletionProtection: un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se per il cluster di database è abilitata o meno la protezione da eliminazione. Il database non può essere eliminato quando è abilitata la protezione da eliminazione.

- EarliestBacktrackTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Non supportato da Neptune.

- EarliestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica il primo orario possibile per il ripristino point-in-time di un database.

- EnabledCloudwatchLogsExports: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Elenco di tipi di log che questo cluster database può esportare in CloudWatch Logs. I tipi di log validi sono: `audit` (per pubblicare log di audit su CloudWatch) e slowquery (per pubblicare log

di slow query su CloudWatch). Consulta Pubblicazione di log Neptune nei file di log Amazon CloudWatch.

- Endpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'endpoint di connessione per l'istanza primaria del cluster di database.

- Engine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il nome del motore di database da utilizzare per questo cluster di database.

- EngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica la versione del motore di database.

- GlobalClusterIdentifier: un valore GlobalClusterIdentifier di tipo `string` (stringa codificata UTF-8), non inferiore a 1 o superiore a 255 ?st?s, corrispondente a questa espressione regolare: `[A-Za-z][0-9A-Za-z-:._]*`.

  Contiene un identificatore del cluster database globale fornito dall'utente. Questo identificatore è la chiave univoca che identifica un database globale.

- HostedZoneId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'ID che Amazon Route 53 assegna al momento della creazione di una zona ospitata.

- IAMDatabaseAuthenticationEnabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.

- IOOptimizedNextAllowedModificationTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La prossima volta puoi modificare il cluster di database per utilizzare il tipo di archiviazione `iopt1`.

- KmsKeyId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per il cluster di database crittografato.

- LatestRestorableTime: un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica l'ultimo orario possibile per il ripristino point-in-time di un database.

- MultiAZ: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database ha istanze in più zone di disponibilità.

- PendingModifiedValues: un oggetto ClusterPendingModifiedValues.

  Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione `ModifyDBCluster` e contiene le modifiche che verranno applicate nella finestra di manutenzione successiva.

- PercentProgress: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'avanzamento dell'operazione sotto forma di percentuale.

- Port: un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto il motore di database.

- PreferredBackupWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'intervallo di tempo giornaliero in cui vengono creati i backup automatici se questi sono abilitati, come determinato da `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica un intervallo temporale settimanale nel fuso orario UTC (Universal Coordinated Time) durante il quale può verificarsi la manutenzione dei sistemi.

- ReaderEndpoint: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'endpoint di lettura per il cluster di database. L'endpoint di lettura per un cluster di database bilancia il carico delle connessioni tra le repliche di lettura disponibili in un cluster di database. Man mano che i client richiedono nuove connessioni all'endpoint di lettura, Neptune distribuisce tali richieste fra le repliche di lettura nel cluster di database. Questa funzionalità è utile per bilanciare il carico di lavoro di lettura tra più repliche di lettura nel cluster di database.

  Se si verifica un failover e la replica di lettura a cui sei connesso viene promossa a istanza primaria, la connessione viene interrotta. Per continuare a inviare il carico di lavoro di lettura ad altre repliche di lettura nel cluster, puoi quindi riconnetterti all'endpoint di lettura.

- ReadReplicaIdentifiers: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene uno o più identificatori delle repliche di lettura associate a questo cluster di database.

- ReplicationSourceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ReplicationType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Non supportato da Neptune.

- ServerlessV2ScalingConfiguration: un oggetto ServerlessV2ScalingConfigurationInfo.

  Mostra la configurazione del dimensionamento per un cluster database Neptune Serverless.

  Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato corrente del cluster di database.

- StorageEncrypted: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se il cluster di database è crittografato.

- StorageType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione utilizzato dal cluster di database.

  Valori validi:

  - **`standard`**: (impostazione predefinita) fornisce archiviazione di database dai costi contenuti per applicazioni con un numero di operazioni di I/O da moderato a basso.
  - **`iopt1`**: abilita l'archiviazione ottimizzata per l'I/O che è progettata per soddisfare le esigenze di carichi di lavoro grafici con un numero elevato di operazioni di I/O che richiedono prezzi prevedibili con bassa latenza I/O e velocità di trasmissione effettiva I/O coerente.

    L'archiviazione ottimizzata per l'I/O di Neptune è disponibile solo a partire dal rilascio 1.3.0.0 del motore.

- VpcSecurityGroups: una matrice di oggetti VpcSecurityGroupMembership.

  Fornisce un elenco dei gruppi di sicurezza VPC a cui appartiene il cluster di database.

Errori

- DBClusterAlreadyExistsFault
- DBClusterNotFoundFault
- DBClusterQuotaExceededFault

- DBClusterSnapshotNotFoundFault
- DBSubnetGroupNotFoundFault
- InsufficientDBClusterCapacityFault
- InsufficientStorageClusterCapacityFault
- InvalidDBClusterSnapshotStateFault
- InvalidDBClusterStateFault
- InvalidDBSnapshotStateFault
- InvalidRestoreFault
- InvalidSubnet
- InvalidVPCNetworkStateFault
- KMSKeyNotAccessibleFault
- OptionGroupNotFoundFault
- StorageQuotaExceededFault
- DBClusterParameterGroupNotFoundFault

DescribeDBClusterSnapshots (operazione)

Il nome AWS CLI per questa API è: `describe-db-cluster-snapshots`.

Restituisce informazioni sugli snapshot del cluster di database. Questa operazione API supporta la paginazione.

Richiesta

- DBClusterIdentifier (nella CLI: `--db-cluster-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID del cluster di database per il quale recuperare l'elenco degli snapshot del cluster di database. Questo parametro non può essere utilizzato in combinazione con il parametro `DBClusterSnapshotIdentifier`. Questo parametro non fa distinzione tra maiuscole e minuscole.

  Vincoli:

  - Se viene specificato, deve corrispondere all'identificatore di un DBCluster esistente.

- DBClusterSnapshotIdentifier (nella CLI: `--db-cluster-snapshot-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un identificatore dello snapshot del cluster di database specifico da descrivere. Questo parametro non può essere utilizzato in combinazione con il parametro `DBClusterIdentifier`. Questo valore è archiviato come stringa in caratteri minuscoli.

  Vincoli:

  - Se viene specificato, deve corrispondere all'identificatore di un DBClusterSnapshot esistente.
  - Se questo identificatore è per uno snapshot automatizzato, anche il parametro `SnapshotType` deve essere specificato.
- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Questo parametro non è attualmente supportato.
- IncludePublic (nella CLI: `--include-public`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True per includere gli snapshot del cluster database manuale che sono pubblici e possono essere copiati o ripristinati da qualsiasi account Amazon, in caso contrario false. Il valore predefinito è `false`. Il valore di default è false.

  È possibile condividere uno snapshot del cluster di database manuale come pubblico utilizzando l'operazione API the section called “ModifyDBClusterSnapshotAttribute”.
- IncludeShared (nella CLI: `--include-shared`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True per includere gli snapshot del cluster database manuale condivisi da altri account Amazon che questo account Amazon è autorizzato a copiare o ripristinare, in caso contrario false. Il valore predefinito è `false`.

  Puoi fornire a un account Amazon l'autorizzazione per ripristinare uno snapshot del cluster database manuale da un altro account Amazon mediante l'azione API the section called “ModifyDBClusterSnapshotAttribute”.
- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta `DescribeDBClusterSnapshots` precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.

- SnapshotType (nella CLI: `--snapshot-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di snapshot del cluster di database da restituire. È possibile specificare uno dei seguenti valori:

  - `automated`: restituisce tutti gli snapshot di cluster database che sono stati eseguiti automaticamente da Amazon Neptune per l'account Amazon personale.
  - `manual`: restituisce tutti gli snapshot di cluster database che sono stati eseguiti dall'account Amazon personale.
  - `shared`: restituisce tutti gli snapshot di cluster database manuale che sono stati condivisi con l'account Amazon personale.
  - `public`: restituisce tutti gli snapshot di cluster database che sono stati contrassegnati come pubblici.

  Se non si specifica un valore `SnapshotType`, entrambi gli snapshot di cluster di database automatici e manuali vengono restituiti. È possibile includere gli snapshot del cluster di database condivisi con questi risultati impostando il parametro `IncludeShared` su `true`. È possibile includere gli snapshot del cluster di database pubblici con questi risultati impostando il parametro `IncludePublic` su `true`.

  I parametri `IncludePublic` e `IncludeShared` non sono applicabili ai valori `SnapshotType` di `manual` o `automated`. Il parametro `IncludePublic` non è applicabile quando `SnapshotType` è impostato su `shared`. Il parametro `IncludeShared` non è applicabile quando `SnapshotType` è impostato su `public`.

Risposta

- DBClusterSnapshots: una matrice di oggetti DBClusterSnapshot.

  Fornisce un elenco di snapshot del cluster di database per l'utente.

- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta the section called "DescribeDBClusterSnapshots" precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

Errori

- DBClusterSnapshotNotFoundFault

DescribeDBClusterSnapshotAttributes (operazione)

Il nome AWS CLI per questa API è: `describe-db-cluster-snapshot-attributes`.

Restituisce un elenco di nomi e valori di attributo dello snapshot del cluster di database per uno snapshot del cluster di database manuale.

Quando si condividono snapshot con altri account Amazon, `DescribeDBClusterSnapshotAttributes` restituisce l'attributo `restore` e un elenco di ID per gli account Amazon che sono autorizzati a copiare o ripristinare lo snapshot del cluster database manuale. Se `all` è incluso nell'elenco di valori per l'attributo `restore`, lo snapshot del cluster database manuale è pubblico e può essere copiato o ripristinato da tutti gli account Amazon.

Per aggiungere o rimuovere l'accesso per consentire a un account Amazon di copiare o ripristinare uno snapshot del cluster database manuale o per rendere lo snapshot del cluster database manuale pubblico o privato, utilizza l'azione API the section called "ModifyDBClusterSnapshotAttribute".

Richiesta

- DBClusterSnapshotIdentifier (nella CLI: `--db-cluster-snapshot-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore per lo snapshot del cluster di database per il quale descrivere gli attributi.

Risposta

Contiene i risultati di una chiamata riuscita all'operazione API the section called “DescribeDBClusterSnapshotAttributes”.

Gli attributi dello snapshot del cluster database manuale vengono utilizzati per autorizzare altri account Amazon a copiare o ripristinare uno snapshot del cluster database manuale. Per ulteriori informazioni, consultare the section called “ModifyDBClusterSnapshotAttribute” operazione API.

- DBClusterSnapshotAttributes: una matrice di oggetti DBClusterSnapshotAttribute.

  L'elenco di attributi e valori per lo snapshot del cluster di database manuale.
- DBClusterSnapshotIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore dello snapshot del cluster di database manuale al quale si applicano gli attributi.

Errori

- DBClusterSnapshotNotFoundFault

Strutture:

DBClusterSnapshot (struttura)

Contiene i dettagli per uno snapshot del cluster di database Amazon Neptune.

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called “DescribeDBClusterSnapshots”.

Campi

- AllocatedStorage: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

  Specifica le dimensioni dello storage allocato, in gibibyte (GiB).
- AvailabilityZones: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'elenco delle zone di disponibilità EC2 nelle quali è possibile ripristinare istanze nello snapshot del cluster di database.
- ClusterCreateTime: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

Specifica l'ora di creazione del cluster di database, nel fuso orario UTC (Universal Coordinated Time).

- DBClusterIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'identificatore del cluster di database, dal quale è stato creato questo snapshot del cluster di database.

- DBClusterSnapshotArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per lo snapshot del cluster di database.

- DBClusterSnapshotIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'identificatore per uno snapshot del cluster DB. Deve corrispondere all'identificatore di una snapshot esistente.

  Dopo aver ripristinato un cluster di database utilizzando `DBClusterSnapshotIdentifier`, è necessario specificare lo stesso `DBClusterSnapshotIdentifier` per eventuali aggiornamenti futuri al cluster di database. Quando specifichi questa proprietà per un aggiornamento, il cluster di database non viene nuovamente ripristinato dallo snapshot e i dati nel database non vengono modificati.

  Tuttavia, se non specifichi `DBClusterSnapshotIdentifier`, viene creato un cluster DB vuoto e il cluster DB originale viene eliminato. Se specifichi una proprietà diversa dalla proprietà di ripristino dello snapshot precedente, il cluster DB viene ripristinato dallo snapshot specificato da `DBClusterSnapshotIdentifier` e il cluster DB originale viene eliminato.

- Engine: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome del motore di database.

- EngineVersion: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce la versione del motore di database da utilizzare per questo snapshot del cluster di database.

- IAMDatabaseAuthenticationEnabled: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  True se la mappatura degli account Amazon Identity and Access Management (IAM) agli account di database è abilitata, in caso contrario false.

- KmsKeyId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

Se `StorageEncrypted` è true, l'identificatore della chiave Amazon KMS per lo snapshot del cluster database crittografato.

- LicenseModel: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce le informazioni del modello di licenza per questo snapshot del cluster di database.

- PercentProgress: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la percentuale dei dati stimati che sono stati trasferiti.

- Port: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la porta con la quale il cluster di database ascoltava al momento della creazione dello snapshot.

- SnapshotCreateTime: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Fornisce l'ora di creazione dello snapshot, nel formato UTC (Universal Coordinated Time).

- SnapshotType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il tipo di snapshot del cluster di database.

- SourceDBClusterSnapshotArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se lo snapshot del cluster di database è stato copiato da uno snapshot del cluster di database di origine, l'Amazon Resource Name (ARN) per lo snapshot del cluster di database di origine, in caso contrario, presenta un valore null.

- Status: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica lo stato dello snapshot del cluster di database.

- StorageEncrypted: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Specifica se lo snapshot del cluster di database è crittografato.

- StorageType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di archiviazione associato allo snapshot del cluster di database.

- VpcId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

Fornisce l'ID VPC associato allo snapshot del cluster di database.

`DBClusterSnapshot` viene utilizzato come elemento di risposta per:

- CreateDBClusterSnapshot
- CopyDBClusterSnapshot
- DeleteDBClusterSnapshot

DBClusterSnapshotAttribute (struttura)

Contiene il nome e i valori di un attributo dello snapshot del cluster di database manuale.

Gli attributi dello snapshot del cluster database manuale vengono utilizzati per autorizzare altri account Amazon a ripristinare uno snapshot del cluster database manuale. Per ulteriori informazioni, consultare the section called “ModifyDBClusterSnapshotAttribute” operazione API.

Campi

- AttributeName: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome dell'attributo dello snapshot del cluster di database manuale.

  L'attributo denominato `restore` fa riferimento all'elenco di account Amazon autorizzati a copiare o ripristinare lo snapshot del cluster database manuale. Per ulteriori informazioni, consultare the section called “ModifyDBClusterSnapshotAttribute” operazione API.
- AttributeValues: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  I valori per l'attributo dello snapshot del cluster di database manuale.

  Se il campo `AttributeName` è impostato su `restore`, questo elemento restituisce un elenco di ID degli account Amazon autorizzati a copiare o ripristinare lo snapshot del cluster database manuale. Se il valore `all` è nell'elenco, lo snapshot del cluster database manuale è pubblico e disponibile per essere copiato o ripristinato da qualsiasi account Amazon.

DBClusterSnapshotAttributesResult (struttura)

Contiene i risultati di una chiamata riuscita all'operazione API the section called “DescribeDBClusterSnapshotAttributes”.

Gli attributi dello snapshot del cluster database manuale vengono utilizzati per autorizzare altri account Amazon a copiare o ripristinare uno snapshot del cluster database manuale. Per ulteriori informazioni, consultare the section called “ModifyDBClusterSnapshotAttribute” operazione API.

Campi

- DBClusterSnapshotAttributes: questo è un array di oggetti DBClusterSnapshotAttribute.

  L'elenco di attributi e valori per lo snapshot del cluster di database manuale.
- DBClusterSnapshotIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore dello snapshot del cluster di database manuale al quale si applicano gli attributi.

`DBClusterSnapshotAttributesResult` viene utilizzato come elemento di risposta per:

- DescribeDBClusterSnapshotAttributes
- ModifyDBClusterSnapshotAttribute

API Events Neptune

Operazioni:

- CreateEventSubscription (operazione)
- DeleteEventSubscription (operazione)
- ModifyEventSubscription (operazione)
- DescribeEventSubscriptions (operazione)
- AddSourceIdentifierToSubscription (operazione)
- RemoveSourceIdentifierFromSubscription (operazione)
- DescribeEvents (operazione)
- DescribeEventCategories (operazione)

Strutture:

- Evento (struttura)
- EventCategoriesMap (struttura)
- EventSubscription (struttura)

CreateEventSubscription (operazione)

Il nome AWS CLI per questa API è: `create-event-subscription`.

Crea una sottoscrizione alle notifiche di eventi. Questa operazione richiede un argomento ARN (Amazon Resource Name) creato mediante la console Neptune, la console SNS oppure l'API SNS. Per ottenere un ARN con SNS, è necessario creare un argomento in Amazon SNS e sottoscriversi all'argomento. L'ARN viene visualizzato nella console SNS.

È possibile specificare il tipo di origine (SourceType) del quale si desidera ricevere una notifica, fornire un elenco di sorgenti Neptune (SourceIds) che attivano gli eventi e forniscono un elenco di categorie di eventi (EventCategories) per gli eventi per i quali si desidera ricevere una notifica. Ad esempio, è possibile specificare SourceType = db-instance, SourceIds = mydbinstance1, mydbinstance2 e EventCategories = Availability, Backup.

Se si specificano sia SourceType sia SourceIds, come SourceType = db-instance e SourceIdentifier = myDBInstance1, si riceve una notifica su tutti gli eventi db-instance per l'origine specificata. Se si specifica un SourceType, ma non un SourceIdentifier, si ricevono le notifiche di eventi per quel tipo di origine per tutte le origini Neptune. Se non si specifica né SourceType né SourceIdentifier, si ricevono le notifiche di eventi generati da tutte le origini Neptune appartenenti all'account cliente.

Richiesta

- Enabled (nella CLI: `--enabled`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore booleano; impostare su true per attivare la sottoscrizione, impostare su false per creare la sottoscrizione ma non attivarla.
- EventCategories (nella CLI: `--event-categories`): una stringa di tipo `string` (una stringa con codifica UTF-8).

Un elenco di categorie di eventi per un SourceType al quale si desidera sottoscriversi. È possibile visualizzare un elenco delle categorie per un determinato SourceType utilizzando l'operazione DescribeEventCategories.

- SnsTopicArn (nella CLI: `--sns-topic-arn`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) dell'argomento SNS creato per la notifica di eventi. L'ARN viene creato da Amazon SNS al momento della creazione di un argomento e la sottoscrizione.

- SourceIds (nella CLI: `--source-ids`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'elenco di identificatori di origini di eventi per le quali vengono restituiti gli eventi. Se non è specificato, tutte le origini sono incluse nella risposta. Un identificare deve iniziare con una lettera e deve contenere solo caratteri ASCII, cifre e trattini, non può terminare con un trattino o contenere due trattini consecutivi.

  Vincoli:

  - Se vengono forniti SourceIds, SourceType deve essere anche fornito.
  - Se il tipo di origine è un'istanza database, un `DBInstanceIdentifier` deve essere fornito.
  - Se il tipo di origine è un gruppo di sicurezza DB, un `DBSecurityGroupName` deve essere fornito.
  - Se il tipo di origine è un gruppo di parametri database, un `DBParameterGroupName` deve essere fornito.
  - Se il tipo di origine è uno snapshot DB, un `DBSnapshotIdentifier` deve essere fornito.

- SourceType (nella CLI: `--source-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di origine che genera gli eventi. Ad esempio, se si desidera ricevere notifiche di eventi generate da un'istanza database, impostare questo parametro su db-instance. Se questo valore non è specificato, tutti gli eventi vengono restituiti.

  Valori validi: `db-instance` | `db-cluster` | `db-parameter-group` | `db-security-group` | `db-snapshot` | `db-cluster-snapshot`

- SubscriptionName (nella CLI: `--subscription-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

Il nome dell'abbonamento.

Vincoli: il nome deve essere inferiore a 255 caratteri.

- Tags (nella CLI: `--tags`): un array di oggetti Tag.

  I tag da applicare alla sottoscrizione al nuovo evento.

Risposta

Contiene i risultati di una invocazione riuscita dell'operazione the section called "DescribeEventSubscriptions".

- CustomerAwsId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'account cliente Amazon associato alla sottoscrizione alle notifiche di eventi.
- CustSubscriptionId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID iscrizione alle notifiche di eventi.
- Enabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore booleano che indica se la sottoscrizione è abilitata. True indica la sottoscrizione è abilitata.
- EventCategoriesList: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco categorie di eventi per l'iscrizione alle notifiche di eventi.
- EventSubscriptionArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per la sottoscrizione all'evento.
- SnsTopicArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'argomento ARN dell'abbonamento delle notifiche di eventi.
- SourceIdsList: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di ID origine per l'iscrizione alle notifiche di eventi.
- SourceType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di origine per l'iscrizione alle notifiche di eventi.
- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

Lo stato della sottoscrizione alle notifiche degli eventi .

Vincoli:

Può essere uno dei seguenti: la creazione di | modificare | l'eliminazione | attivi | senza permesso | topic-not-esistenti

Lo stato "no-permission" indica che Neptune non ha più l'autorizzazione di pubblicare nell'argomento SNS. Lo stato "topic-not-exist" indica che l'argomento è stato eliminato dopo la creazione dell'iscrizione.

- SubscriptionCreationTime: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ora di creazione della sottoscrizione alle notifiche di eventi.

Errori

- EventSubscriptionQuotaExceededFault
- SubscriptionAlreadyExistFault
- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault
- SubscriptionCategoryNotFoundFault
- SourceNotFoundFault

DeleteEventSubscription (operazione)

Il nome AWS CLI per questa API è: `delete-event-subscription`.

Elimina una sottoscrizione alle notifiche di eventi.

Richiesta

- SubscriptionName (nella CLI: `--subscription-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della sottoscrizione alle notifiche di eventi che si desidera eliminare.

Risposta

Contiene i risultati di una invocazione riuscita dell'operazione the section called “DescribeEventSubscriptions”.

- CustomerAwsId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'account cliente Amazon associato alla sottoscrizione alle notifiche di eventi.
- CustSubscriptionId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID iscrizione alle notifiche di eventi.
- Enabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore booleano che indica se la sottoscrizione è abilitata. True indica la sottoscrizione è abilitata.
- EventCategoriesList: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco categorie di eventi per l'iscrizione alle notifiche di eventi.
- EventSubscriptionArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per la sottoscrizione all'evento.
- SnsTopicArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'argomento ARN dell'abbonamento delle notifiche di eventi.
- SourceIdsList: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di ID origine per l'iscrizione alle notifiche di eventi.
- SourceType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di origine per l'iscrizione alle notifiche di eventi.
- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato della sottoscrizione alle notifiche degli eventi .

  Vincoli:

  Può essere uno dei seguenti: la creazione di | modificare | l'eliminazione | attivi | senza permesso | topic-not-esistenti

Lo stato "no-permission" indica che Neptune non ha più l'autorizzazione di pubblicare nell'argomento SNS. Lo stato "topic-not-exist" indica che l'argomento è stato eliminato dopo la creazione dell'iscrizione.

- SubscriptionCreationTime: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ora di creazione della sottoscrizione alle notifiche di eventi.

Errori

- SubscriptionNotFoundFault
- InvalidEventSubscriptionStateFault

ModifyEventSubscription (operazione)

Il nome AWS CLI per questa API è: `modify-event-subscription`.

Modifica una sottoscrizione alle notifiche di eventi esistente. Tenere presente che non è possibile modificare gli identificatori di origine utilizzando questa chiamata; per modificare gli identificatori di origine per una sottoscrizione, utilizzare le chiamate the section called "AddSourceIdentifierToSubscription" e the section called "RemoveSourceIdentifierFromSubscription".

È possibile visualizzare un elenco delle categorie di eventi per un determinato SourceType utilizzando l'operazione DescribeEventCategories.

Richiesta

- Enabled (nella CLI: `--enabled`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore booleano; impostato su true per attivare la sottoscrizione.
- EventCategories (nella CLI: `--event-categories`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di categorie di eventi per un SourceType al quale si desidera sottoscriversi. È possibile visualizzare un elenco delle categorie per un determinato SourceType utilizzando l'operazione DescribeEventCategories.

- SnsTopicArn (nella CLI: `--sns-topic-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) dell'argomento SNS creato per la notifica di eventi. L'ARN viene creato da Amazon SNS al momento della creazione di un argomento e la sottoscrizione.
- SourceType (nella CLI: `--source-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di origine che genera gli eventi. Ad esempio, se si desidera ricevere notifiche di eventi generate da un'istanza database, impostare questo parametro su db-instance. Se questo valore non è specificato, tutti gli eventi vengono restituiti.

  Valori validi: db-instance | db-parameter-group | db-security-group | db-snapshot
- SubscriptionName (nella CLI: `--subscription-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della sottoscrizione alle notifiche degli eventi .

Risposta

Contiene i risultati di una invocazione riuscita dell'operazione the section called “DescribeEventSubscriptions”.

- CustomerAwsId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'account cliente Amazon associato alla sottoscrizione alle notifiche di eventi.
- CustSubscriptionId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID iscrizione alle notifiche di eventi.
- Enabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore booleano che indica se la sottoscrizione è abilitata. True indica la sottoscrizione è abilitata.
- EventCategoriesList: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco categorie di eventi per l'iscrizione alle notifiche di eventi.
- EventSubscriptionArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per la sottoscrizione all'evento.

- SnsTopicArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'argomento ARN dell'abbonamento delle notifiche di eventi.
- SourceIdsList: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di ID origine per l'iscrizione alle notifiche di eventi.
- SourceType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di origine per l'iscrizione alle notifiche di eventi.
- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato della sottoscrizione alle notifiche degli eventi .

  Vincoli:

  Può essere uno dei seguenti: la creazione di | modificare | l'eliminazione | attivi | senza permesso | topic-not-esistenti

  Lo stato "no-permission" indica che Neptune non ha più l'autorizzazione di pubblicare nell'argomento SNS. Lo stato "topic-not-exist" indica che l'argomento è stato eliminato dopo la creazione dell'iscrizione.
- SubscriptionCreationTime: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ora di creazione della sottoscrizione alle notifiche di eventi.

Errori

- EventSubscriptionQuotaExceededFault
- SubscriptionNotFoundFault
- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault
- SubscriptionCategoryNotFoundFault

DescribeEventSubscriptions (operazione)

Il nome AWS CLI per questa API è: `describe-event-subscriptions`.

Elenca tutte le descrizioni di sottoscrizioni per un account cliente. La descrizione di una sottoscrizione include SubscriptionName, SNSTopicARN, CustomerID, SourceType, SourceID, CreationTime e Status.

Se si specifica un SubscriptionName, elenca la descrizione di quell'abbonamento.

Richiesta

- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Questo parametro non è attualmente supportato.
- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta DescribeOrderableDBInstanceOptions precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.
- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.
- SubscriptionName (nella CLI: `--subscription-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della sottoscrizione alle notifiche di eventi che si desidera descrivere.

Risposta

- EventSubscriptionsList: una matrice di oggetti Abbonamento a eventi.

  Un elenco di tipi di dati EventSubscriptions.
- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

Token di paginazione opzionale fornito da una richiesta DescribeOrderableDBInstanceOptions precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

Errori

- SubscriptionNotFoundFault

AddSourceIdentifierToSubscription (operazione)

Il nome AWS CLI per questa API è: `add-source-identifier-to-subscription`.

Aggiunge un identificatore di origine a una sottoscrizione alle notifiche di eventi esistente.

Richiesta

- SourceIdentifier (nella CLI: `--source-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore dell'origine evento da aggiungere.

  Vincoli:

  - Se il tipo di origine è un'istanza database, un `DBInstanceIdentifier` deve essere fornito.
  - Se il tipo di origine è un gruppo di sicurezza DB, un `DBSecurityGroupName` deve essere fornito.
  - Se il tipo di origine è un gruppo di parametri database, un `DBParameterGroupName` deve essere fornito.
  - Se il tipo di origine è uno snapshot DB, un `DBSnapshotIdentifier` deve essere fornito.
- SubscriptionName (nella CLI: `--subscription-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della sottoscrizione alle notifiche di eventi, alla quale si desidera aggiungere un identificatore di origine.

Risposta

Contiene i risultati di una invocazione riuscita dell'operazione the section called “DescribeEventSubscriptions”.

- CustomerAwsId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'account cliente Amazon associato alla sottoscrizione alle notifiche di eventi.

- CustSubscriptionId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID iscrizione alle notifiche di eventi.

- Enabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore booleano che indica se la sottoscrizione è abilitata. True indica la sottoscrizione è abilitata.

- EventCategoriesList: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco categorie di eventi per l'iscrizione alle notifiche di eventi.

- EventSubscriptionArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per la sottoscrizione all'evento.

- SnsTopicArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'argomento ARN dell'abbonamento delle notifiche di eventi.

- SourceIdsList: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di ID origine per l'iscrizione alle notifiche di eventi.

- SourceType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di origine per l'iscrizione alle notifiche di eventi.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato della sottoscrizione alle notifiche degli eventi .

  Vincoli:

  Può essere uno dei seguenti: la creazione di | modificare | l'eliminazione | attivi | senza permesso | topic-not-esistenti

Lo stato "no-permission" indica che Neptune non ha più l'autorizzazione di pubblicare nell'argomento SNS. Lo stato "topic-not-exist" indica che l'argomento è stato eliminato dopo la creazione dell'iscrizione.

- SubscriptionCreationTime: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ora di creazione della sottoscrizione alle notifiche di eventi.

Errori

- SubscriptionNotFoundFault
- SourceNotFoundFault

RemoveSourceIdentifierFromSubscription (operazione)

Il nome AWS CLI per questa API è: `remove-source-identifier-from-subscription`.

Rimuove un identificatore di origine da una sottoscrizione alle notifiche eventi esistente.

Richiesta

- SourceIdentifier (nella CLI: `--source-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore di origine da rimuovere dalla sottoscrizione, ad esempio l'identificatore istanze DB per un'istanza database o il nome di un gruppo di sicurezza.
- SubscriptionName (nella CLI: `--subscription-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della sottoscrizione alle notifiche di eventi, dalla quale si desidera rimuovere un identificatore di origine.

Risposta

Contiene i risultati di una invocazione riuscita dell'operazione the section called “DescribeEventSubscriptions”.

- CustomerAwsId: una stringa di tipo `string` (una stringa con codifica UTF-8).

L'account cliente Amazon associato alla sottoscrizione alle notifiche di eventi.

- CustSubscriptionId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID iscrizione alle notifiche di eventi.

- Enabled: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore booleano che indica se la sottoscrizione è abilitata. True indica la sottoscrizione è abilitata.

- EventCategoriesList: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco categorie di eventi per l'iscrizione alle notifiche di eventi.

- EventSubscriptionArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per la sottoscrizione all'evento.

- SnsTopicArn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'argomento ARN dell'abbonamento delle notifiche di eventi.

- SourceIdsList: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di ID origine per l'iscrizione alle notifiche di eventi.

- SourceType: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di origine per l'iscrizione alle notifiche di eventi.

- Status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato della sottoscrizione alle notifiche degli eventi .

  Vincoli:

  Può essere uno dei seguenti: la creazione di | modificare | l'eliminazione | attivi | senza permesso | topic-not-esistenti

  Lo stato "no-permission" indica che Neptune non ha più l'autorizzazione di pubblicare nell'argomento SNS. Lo stato "topic-not-exist" indica che l'argomento è stato eliminato dopo la creazione dell'iscrizione.

- SubscriptionCreationTime: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ora di creazione della sottoscrizione alle notifiche di eventi.

Errori

- SubscriptionNotFoundFault
- SourceNotFoundFault

DescribeEvents (operazione)

Il nome AWS CLI per questa API è: `describe-events`.

Restituisce eventi relativi a istanze database, gruppi di sicurezza DB , snapshot DB e gruppi di parametri database per gli ultimi 14 giorni. Gli eventi relativi a istanze database, gruppi di sicurezza DB, snapshot DB o gruppo di parametri database determinati possono essere ottenuti fornendo il nome come un parametro. Come impostazione predefinita, viene restituita l'ultima ora degli eventi.

Richiesta

- Duration (nella CLI: `--duration`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Il numero di minuti per il quale recuperare gli eventi.

  Impostazione predefinita: 60
- EndTime (nella CLI: `--end-time`): un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La fine dell'intervallo di tempo per il quale recuperare gli eventi, specificato nel formato ISO 8601. Per ulteriori informazioni su ISO 8601, consultare la pagina Wikepedia ISO8601.

  Esempio: 2009-07-08T18:00Z
- EventCategories (nella CLI: `--event-categories`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di categorie di eventi che attivano le notifiche per una sottoscrizione alle notifiche di eventi.
- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Questo parametro non è attualmente supportato.
- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

Un token di paginazione opzionale fornito da una richiesta DescribeEvents precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.

- SourceIdentifier (nella CLI: `--source-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore dell'origine dell'evento in base a cui vengono restituiti gli eventi. Se non è specificato, tutte le origini sono incluse nella risposta.

  Vincoli:

  - Se viene fornito SourceIdentifier, SourceType deve essere anche fornito.
  - Se il tipo di origine è `DBInstance`, un `DBInstanceIdentifier` deve essere fornito.
  - Se il tipo di origine è `DBSecurityGroup`, un `DBSecurityGroupName` deve essere fornito.
  - Se il tipo di origine è `DBParameterGroup`, un `DBParameterGroupName` deve essere fornito.
  - Se il tipo di origine è `DBSnapshot`, un `DBSnapshotIdentifier` deve essere fornito.
  - Non può terminare con un trattino o contenere due trattini consecutivi.

- SourceType (nella CLI: `--source-type`): un valore SourceType di tipo `string` (una stringa con codifica UTF-8).

  L'origine eventi per la quale recuperare gli eventi. Se non viene specificato alcun valore, tutti gli eventi vengono restituiti.

- StartTime (nella CLI: `--start-time`): un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  L'inizio dell'intervallo di tempo per il quale recuperare gli eventi, specificato nel formato ISO 8601. Per ulteriori informazioni su ISO 8601, consultare la pagina Wikepedia ISO8601.

Esempio: 2009-07-08T18:00Z

Risposta

- Events: una matrice di oggetti Evento.

  Elenco di istanze di the section called “Evento”.

- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta Events precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

DescribeEventCategories (operazione)

Il nome AWS CLI per questa API è: `describe-event-categories`.

Visualizza un elenco di categorie per tutti i tipi di origini di eventi, oppure, se specificato, per un determinato tipo di origine.

Richiesta

- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Questo parametro non è attualmente supportato.

- SourceType (nella CLI: `--source-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di origine che genera gli eventi.

  Valori validi: db-instance | db-parameter-group | db-security-group | db-snapshot

Risposta

- EventCategoriesMapList: una matrice di oggetti EventCategoriesMap.

  Un elenco di tipi di dati EventCategoriesMap.

Strutture:

Evento (struttura)

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called “DescribeEvents”.

Campi

- Date: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  Specifica la data e l'ora dell'evento.
- EventCategories: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica la categoria per l'evento.
- Message: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce il testo di questo evento.
- SourceArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per l'evento.
- SourceIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Fornisce l'identificatore per l'origine dell'evento.
- SourceType: questo è un valore SourceType di tipo `string` (una stringa con codifica UTF-8).

  Specifica il tipo di origine per questo evento.

EventCategoriesMap (struttura)

Contiene i risultati di una invocazione riuscita dell'operazione the section called “DescribeEventCategories”.

Campi

- EventCategories: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Le categorie di eventi per il tipo di origine specificato

- SourceType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di origine al quale appartengono le categorie

EventSubscription (struttura)

Contiene i risultati di una invocazione riuscita dell'operazione the section called "DescribeEventSubscriptions".

Campi

- CustomerAwsId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'account cliente Amazon associato alla sottoscrizione alle notifiche di eventi.
- CustSubscriptionId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID iscrizione alle notifiche di eventi.
- Enabled: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore booleano che indica se la sottoscrizione è abilitata. True indica la sottoscrizione è abilitata.
- EventCategoriesList: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco categorie di eventi per l'iscrizione alle notifiche di eventi.
- EventSubscriptionArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) per la sottoscrizione all'evento.
- SnsTopicArn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'argomento ARN dell'abbonamento delle notifiche di eventi.
- SourceIdsList: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di ID origine per l'iscrizione alle notifiche di eventi.
- SourceType: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di origine per l'iscrizione alle notifiche di eventi.
- Status: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato della sottoscrizione alle notifiche degli eventi .

Vincoli:

Può essere uno dei seguenti: la creazione di | modificare | l'eliminazione | attivi | senza permesso | topic-not-esistenti

Lo stato "no-permission" indica che Neptune non ha più l'autorizzazione di pubblicare nell'argomento SNS. Lo stato "topic-not-exist" indica che l'argomento è stato eliminato dopo la creazione dell'iscrizione.

- SubscriptionCreationTime: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ora di creazione della sottoscrizione alle notifiche di eventi.

`EventSubscription` viene utilizzato come elemento di risposta per:

- CreateEventSubscription
- ModifyEventSubscription
- AddSourceIdentifierToSubscription
- RemoveSourceIdentifierFromSubscription
- DeleteEventSubscription

Altre API Neptune

Operazioni:

- AddTagsToResource (operazione)
- ListTagsForResource (operazione)
- RemoveTagsFromResource (operazione)
- ApplyPendingMaintenanceAction (operazione)
- DescribePendingMaintenanceActions (operazione)
- DescribeDBEngineVersions (operazione)

Strutture:

- DBEngineVersion (struttura)

- EngineDefaults (struttura)
- PendingMaintenanceAction (struttura)
- ResourcePendingMaintenanceActions (struttura)
- UpgradeTarget (struttura)
- Tag (struttura)

AddTagsToResource (operazione)

Il nome AWS CLI per questa API è: `add-tags-to-resource`.

Aggiunge tag di metadati a una risorsa Amazon Neptune. Questi tag possono anche essere utilizzati con la creazione di rapporti sull'allocazione dei costi per monitorare i costi associati alle risorse Amazon Neptune o utilizzati in un'istruzione di condizione in una policy IAM per Amazon Neptune.

Richiesta

- ResourceName (nella CLI: `--resource-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La risorsa Amazon Neptune alla quale vengono aggiunti i tag. Questo valore è un Amazon Resource Name (ARN). Per informazioni su come creare un ARN, consultare Costruzione di un ARN (Amazon Resource Name).
- Tags (nella CLI: `--tags`): Obbligatorio: un array di oggetti Tag.

  I tag da assegnare alla risorsa Amazon Neptune.

Risposta

- Nessun parametro di risposta.

Errori

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

ListTagsForResource (operazione)

Il nome AWS CLI per questa API è: `list-tags-for-resource`.

Elenca tutti i tag su una risorsa Amazon Neptune.

Richiesta

- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Questo parametro non è attualmente supportato.

- ResourceName (nella CLI: `--resource-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La risorsa Amazon Neptune con tag da elencare. Questo valore è un Amazon Resource Name (ARN). Per informazioni su come creare un ARN, consultare Costruzione di un ARN (Amazon Resource Name).

Risposta

- TagList: una matrice di oggetti Tag.

  Elenco dei tag restituiti dall'operazione ListTagsForResource.

Errori

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

RemoveTagsFromResource (operazione)

Il nome AWS CLI per questa API è: `remove-tags-from-resource`.

Rimuove tag di metadati da una risorsa Amazon Neptune.

Richiesta

- ResourceName (nella CLI: `--resource-name`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La risorsa Amazon Neptune dalla quale vengono rimossi i tag. Questo valore è un Amazon Resource Name (ARN). Per informazioni su come creare un ARN, consultare Costruzione di un ARN (Amazon Resource Name).
- TagKeys (nella CLI: `--tag-keys`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La chiave del tag (nome) del tag da rimuovere.

Risposta

- Nessun parametro di risposta.

Errori

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

ApplyPendingMaintenanceAction (operazione)

Il nome AWS CLI per questa API è: `apply-pending-maintenance-action`.

Applica un'operazione di manutenzione in sospeso a una risorsa (ad esempio, a un'istanza database).

Richiesta

- ApplyAction (nella CLI: `--apply-action`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'operazione di manutenzione in sospeso da applicare a questa risorsa.

  Valori validi: `system-update`, `db-upgrade`
- OptInType (nella CLI: `--opt-in-type`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

Un valore che specifica il tipo di richiesta di consenso esplicito o ne annulla una. Una richiesta di consenso esplicito di tipo `immediate` non può essere annullata.

Valori validi:

- `immediate`: applica immediatamente l'azione di manutenzione.
- `next-maintenance` -Applica l'operazione di manutenzione durante la finestra di manutenzione successiva per la risorsa.
- `undo-opt-in`: annulla qualsiasi richiesta di consenso esplicito `next-maintenance` esistente.

- ResourceIdentifier (nella CLI: `--resource-identifier`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'Amazon Resource Name (ARN) della risorsa alla quale viene applicata l'operazione di manutenzione in sospeso. Per informazioni su come creare un ARN, consultare Costruzione di un ARN (Amazon Resource Name).

Risposta

Descrive le operazioni di manutenzione in sospeso per una risorsa.

- PendingMaintenanceActionDetails: una matrice di oggetti PendingMaintenanceAction.

  Un elenco che fornisce i dettagli sulle operazioni di manutenzione in sospeso per la risorsa.
- ResourceIdentifier: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La ARN della risorsa che dispone di operazioni di manutenzione in sospeso.

Errori

- ResourceNotFoundFault

DescribePendingMaintenanceActions (operazione)

Il nome AWS CLI per questa API è: `describe-pending-maintenance-actions`.

Restituisce un elenco di risorse (ad esempio istanze database) che hanno almeno un'operazione di manutenzione in sospeso.

Richiesta

- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Un filtro che specifica una o più risorse per restituire le operazioni di manutenzione in sospeso.

  Filtri supportati:

  - `db-cluster-id`: accetta identificatori e Amazon Resource Name (ARN) di cluster database. L'elenco di risultati include solo le operazioni di manutenzione in sospeso per i cluster di database identificati da questi ARN.
  - `db-instance-id`: accetta gli identificatori di istanze database e gli ARN di istanze database. L'elenco di risultati include solo le operazioni di manutenzione in sospeso per le istanze DB identificate da questi ARN.
- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta `DescribePendingMaintenanceActions` precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino a un numero di record specificato da `MaxRecords`.
- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più record rispetto al valore di `MaxRecords` specificato, nella risposta viene incluso un token di paginazione detto contrassegno (oggetto Marker), per permettere di recuperare i risultati rimanenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.
- ResourceIdentifier (nella CLI: `--resource-identifier`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN della risorsa per la quale restituire operazioni di manutenzione in sospeso.

Risposta

- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

Token di paginazione opzionale fornito da una richiesta `DescribePendingMaintenanceActions` precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino a un numero di record specificato da `MaxRecords`.

- PendingMaintenanceActions: una matrice di oggetti ResourcePendingMaintenanceActions.

  Un elenco di operazioni di manutenzione in sospeso per la risorsa.

Errori

- ResourceNotFoundFault

DescribeDBEngineVersions (operazione)

Il nome AWS CLI per questa API è: `describe-db-engine-versions`.

Restituisce un elenco dei motori di database disponibili.

Richiesta

- DBParameterGroupFamily (nella CLI: `--db-parameter-group-family`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome di una determinato famiglia di gruppo di parametri database per la quale restituire i dettagli.

  Vincoli:

  - Se viene specificato, deve corrispondere a un oggetto DBParameterGroupFamily esistente.
- DefaultOnly (nella CLI: `--default-only`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica che solo la versione predefinita del motore specificato o una combinazione di motore e versione principale viene restituita.
- Engine (nella CLI: `--engine`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il motore di database da restituire.
- EngineVersion (nella CLI: `--engine-version`): una stringa di tipo `string` (una stringa con codifica UTF-8).

La versione del motore di database da restituire.

Esempio: `5.1.49`

- Filters (nella CLI: `--filters`): un array di oggetti Filtro.

  Attualmente non è supportato.

- ListSupportedCharacterSets (nella CLI: `--list-supported-character-sets`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se questo parametro viene specificato e il modulo di richiesta supporta il parametro `CharacterSetName` per `CreateDBInstance`, la risposta include un elenco di set di caratteri supportati per ciascuna versione del motore.

- ListSupportedTimezones (nella CLI: `--list-supported-timezones`): un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Se questo parametro viene specificato e il modulo di richiesta supporta il parametro `TimeZone` per `CreateDBInstance`, la risposta include un elenco di fusi orari supportati per ciascuna versione del motore.

- Marker (nella CLI: `--marker`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

- MaxRecords (nella CLI: `--max-records`): un valore IntegerOptional di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di record da includere nella risposta. Se sono presenti più valori oltre a `MaxRecords`, nella risposta viene incluso un token di paginazione detto contrassegno, per permettere di recuperare i risultati seguenti.

  Impostazione predefinita: 100

  Vincoli: minimo 20, massimo 100.

Risposta

- DBEngineVersions: una matrice di oggetti DBEngineVersion.

Un elenco di elementi `DBEngineVersion`.

- Marker: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Token di paginazione opzionale fornito da una richiesta precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

Strutture:

DBEngineVersion (struttura)

Questo tipo di dati viene utilizzato come elemento di risposta nell'operazione the section called "DescribeDBEngineVersions".

Campi

- DBEngineDescription: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  La descrizione del motore di database.

- DBEngineVersionDescription: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  La descrizione della versione del motore di database.

- DBParameterGroupFamily: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della famiglia del gruppo di parametri database per il motore di database.

- Engine: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del motore di database.

- EngineVersion: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il numero di versione del motore di database.

- ExportableLogTypes: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  I tipi di log che il motore di database ha a disposizione per l'esportazione in CloudWatch Logs.

- SupportedTimezones: questo è un array di oggetti Fuso orario.

Un elenco di fusi orari supportati da questo motore per il parametro `Timezone` dell'operazione `CreateDBInstance`.

- SupportsGlobalDatabases: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se è possibile utilizzare i database globali Aurora con una versione specifica del motore di database.

- SupportsLogExportsToCloudwatchLogs: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se la versione del motore supporta l'esportazione dei tipi di log specificati da ExportableLogTypes in CloudWatch Logs.

- SupportsReadReplica: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se la versione del motore di database supporta le repliche di lettura.

- ValidUpgradeTarget: questo è un array di oggetti UpgradeTarget.

  Un elenco di versioni dei motori alle quali questa versione del motore di database può essere aggiornata.

EngineDefaults (struttura)

Contiene il risultato di una chiamata di successo dell'operazione the section called "DescribeEngineDefaultParameters".

Campi

- DBParameterGroupFamily: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica il nome della famiglia del gruppo di parametri database alla quale si applicano i parametri predefiniti del motore.

- Marker: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un token di paginazione opzionale fornito da una richiesta EngineDefaults precedente. Se questo parametro viene specificato, la risposta include solo i record oltre il contrassegno, fino al valore specificato da `MaxRecords`.

- Parameters: questo è un array di oggetti Parametro.

  Contiene un elenco di parametri predefiniti del motore.

`EngineDefaults` viene utilizzato come elemento di risposta per:

- DescribeEngineDefaultParameters
- DescribeEngineDefaultClusterParameters

PendingMaintenanceAction (struttura)

Fornisce informazioni su un'operazione di manutenzione in sospeso per una risorsa.

Campi

- Action: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di operazione di manutenzione in sospeso che è disponibile per la risorsa.
- AutoAppliedAfterDate: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La data della finestra di manutenzione quando l'operazione viene applicata. L'operazione di manutenzione viene applicata alla risorsa durante la prima finestra di manutenzione dopo questa data. Se questa data è specificata, qualsiasi richiesta di consenso esplicito `next-maintenance` viene ignorata.
- CurrentApplyDate: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

  La data di validità quando l'operazione di manutenzione in sospeso viene applicata alla risorsa. Questa data tiene conto delle richieste di consenso esplicito ricevute dall'API the section called “ApplyPendingMaintenanceAction”, `AutoAppliedAfterDate` e `ForcedApplyDate`. Questo valore è vuoto se una richiesta di consenso esplicito non è stato ricevuta e non è stato specificato nulla come `AutoAppliedAfterDate` o `ForcedApplyDate`.
- Description: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Una descrizione che fornisce dettagli sull'operazione di manutenzione.
- ForcedApplyDate: questo è un valore TStamp di tipo `timestamp` (un punto temporale, generalmente definito come un offset dalla mezzanotte del 01/01/1970).

La data quando l'operazione di manutenzione viene applicata automaticamente. L'operazione di manutenzione viene applicata alla risorsa in questa data indipendentemente dalla finestra di manutenzione per la risorsa. Se questa data è specificata, qualsiasi richiesta di consenso esplicito `immediate` viene ignorata.

- OptInStatus: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Indica il tipo di richiesta di consenso esplicito che è stata ricevuta per la risorsa.

ResourcePendingMaintenanceActions (struttura)

Descrive le operazioni di manutenzione in sospeso per una risorsa.

Campi

- PendingMaintenanceActionDetails: questo è un array di oggetti PendingMaintenanceAction.

  Un elenco che fornisce i dettagli sulle operazioni di manutenzione in sospeso per la risorsa.
- ResourceIdentifier: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  La ARN della risorsa che dispone di operazioni di manutenzione in sospeso.

`ResourcePendingMaintenanceActions` viene utilizzato come elemento di risposta per:

- ApplyPendingMaintenanceAction

UpgradeTarget (struttura)

La versione del motore di database alla quale può essere aggiornata un'istanza database.

Campi

- AutoUpgrade: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se la versione di destinazione viene applicata a qualsiasi istanza database di origine con AutoMinorVersionUpgrade impostato su "true".
- Description: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  La versione del motore di database alla quale può essere aggiornata un'istanza database.

- Engine: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del motore di database di destinazione di aggiornamento.
- EngineVersion: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il numero di versione del motore di database di destinazione di aggiornamento.
- IsMajorVersionUpgrade: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se un motore di database viene aggiornato a una versione principale.
- SupportsGlobalDatabases: questo è un valore BooleanOptional di tipo `boolean` [un valore booleano (vero o falso)].

  Un valore che indica se è possibile utilizzare i database globali Neptune con la versione del motore di destinazione.

Tag (struttura)

I metadati assegnati a una risorsa Amazon Neptune costituita da una coppia chiave-valore.

Campi

- Key: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Una chiave corrisponde al nome obbligatorio del tag. Il valore della stringa può essere composto da 1 a 128 caratteri Unicode e non può avere il prefisso `aws:` o `rds:`. La stringa può contenere solo il set di lettere, cifre, spazi vuoti Unicode, '_', '.', '/', '=', '+', '-' (espressioni regolari Java: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$").
- Value: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un valore è il valore opzione del tag. Il valore della stringa può essere composto da 1 a 256 caratteri Unicode e non può avere il prefisso `aws:` o `rds:`. La stringa può contenere solo il set di lettere, cifre, spazi vuoti Unicode, '_', '.', '/', '=', '+', '-' (espressioni regolari Java: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$").

Tipi di dati comuni di Neptune

Strutture:

AvailabilityZone (struttura)

Specifica una zona di disponibilità.

Campi

- Name: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome della zona di disponibilità.

DBSecurityGroupMembership (struttura)

Specifica l'appartenenza a un gruppo di sicurezza di database designato.

Campi

- DBSecurityGroupName: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del gruppo di sicurezza di database.
- Status: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Stato del gruppo di sicurezza di database.

DomainMembership (struttura)

Un record di appartenenza al dominio Active Directory associato a un'istanza database.

Campi

- Domain: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore del dominio Active Directory.
- FQDN: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome di dominio completo del dominio Active Directory.
- IAMRoleName: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del ruolo IAM da utilizzare per le chiamate API al servizio di directory.
- Status: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Stato di appartenenza al dominio Active Directory dell'istanza database, ad esempio associata, associazione in sospeso, associazione non riuscita e così via.

DoubleRange (struttura)

Intervallo di valori doppi.

Campi

- From: questo è un valore Double di tipo `double` (un numero a virgola mobile IEEE 754 a doppia precisione).

  Valore minimo dell'intervallo.
- To: questo è un valore Double di tipo `double` (un numero a virgola mobile IEEE 754 a doppia precisione).

  Valore massimo dell'intervallo.

Endpoint (struttura)

Specifica un endpoint di connessione.

Per la struttura dei dati che rappresenta gli endpoint del cluster database Amazon Neptune, consulta `DBClusterEndpoint`.

Campi

- Address: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'indirizzo DNS dell'istanza database.
- HostedZoneId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Specifica l'ID che Amazon Route 53 assegna al momento della creazione di una zona ospitata.
- Port: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

  Specifica la porta su cui è in ascolto il motore di database.

Filter (struttura)

Questo tipo non è attualmente supportato.

Campi

- Name: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Questo parametro non è attualmente supportato.
- Values: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Questo parametro non è attualmente supportato.

Range (struttura)

Intervallo di valori interi.

Campi

- From: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

  Valore minimo dell'intervallo.
- Step: questo è un valore IntegerOptional di tipo `integer`(numero intero a 32 bit con segno).

Il valore di incremento per l'intervallo. Ad esempio, nel caso di un intervallo compreso tra 5.000 e 10.000, con un valore di incremento di 1.000, i valori validi partono da 5.000 e incrementano di 1.000. Anche se 7.500 si trova all'interno dell'intervallo, non è un valore valido per l'intervallo. I valori validi sono 5.000, 6.000, 7.000, 8.000...

- To: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

  Valore massimo dell'intervallo.

ServerlessV2ScalingConfiguration (struttura)

Contiene la configurazione di dimensionamento di un cluster database Neptune Serverless.

Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

Campi

- MaxCapacity: questo è un valore DoubleOptional di tipo `double` (un numero a virgola mobile IEEE 754 a doppia precisione).

  Il numero massimo di unità di capacità di Neptune (NCU) per un'istanza database in un cluster Neptune Serverless. Puoi specificare valori NCU con incrementi di mezza unità, ad esempio 40, 40,5, 41 e così via.
- MinCapacity: questo è un valore DoubleOptional di tipo `double` (un numero a virgola mobile IEEE 754 a doppia precisione).

  Il numero minimo di unità di capacità di Neptune (NCU) per un'istanza database in un cluster Neptune Serverless. Puoi specificare valori NCU con incrementi di mezza unità, ad esempio 8, 8,5, 9 e così via.

ServerlessV2ScalingConfigurationInfo (struttura)

Mostra la configurazione del dimensionamento per un cluster database Neptune Serverless.

Per ulteriori informazioni, consulta Utilizzo di Amazon Neptune Serverless nella Guida per l'utente di Amazon Neptune.

Campi

- MaxCapacity: questo è un valore DoubleOptional di tipo `double` (un numero a virgola mobile IEEE 754 a doppia precisione).

  Il numero massimo di unità di capacità di Neptune (NCU) per un'istanza database in un cluster Neptune Serverless. Puoi specificare valori NCU con incrementi di mezza unità, ad esempio 40, 40,5, 41 e così via.
- MinCapacity: questo è un valore DoubleOptional di tipo `double` (un numero a virgola mobile IEEE 754 a doppia precisione).

  Il numero minimo di unità di capacità di Neptune (NCU) per un'istanza database in un cluster Neptune Serverless. Puoi specificare valori NCU con incrementi di mezza unità, ad esempio 8, 8,5, 9 e così via.

Timezone (struttura)

Fuso orario associato a una the section called “DBInstance”.

Campi

- TimezoneName: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del fuso orario.

VpcSecurityGroupMembership (struttura)

Questo tipo di dati viene usato come elemento di risposta per le query sull'appartenenza ai gruppi di sicurezza VPC.

Campi

- Status: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Stato del gruppo di sicurezza VPC.
- VpcSecurityGroupId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Nome del gruppo di sicurezza VPC.

Le eccezioni Neptune specifiche per le API individuali

Eccezioni:

- AuthorizationAlreadyExistsFault (struttura)
- AuthorizationNotFoundFault (struttura)
- AuthorizationQuotaExceededFault (struttura)
- CertificateNotFoundFault (struttura)
- DBClusterAlreadyExistsFault (struttura)
- DBClusterNotFoundFault (struttura)
- DBClusterParameterGroupNotFoundFault (struttura)
- DBClusterQuotaExceededFault (struttura)
- DBClusterRoleAlreadyExistsFault (struttura)
- DBClusterRoleNotFoundFault (struttura)
- DBClusterRoleQuotaExceededFault (struttura)
- DBClusterSnapshotAlreadyExistsFault (struttura)
- DBClusterSnapshotNotFoundFault (struttura)
- DBInstanceAlreadyExistsFault (struttura)
- DBInstanceNotFoundFault (struttura)
- DBLogFileNotFoundFault (struttura)
- DBParameterGroupAlreadyExistsFault (struttura)
- DBParameterGroupNotFoundFault (struttura)
- DBParameterGroupQuotaExceededFault (struttura)
- DBSecurityGroupAlreadyExistsFault (struttura)
- DBSecurityGroupNotFoundFault (struttura)
- DBSecurityGroupNotSupportedFault (struttura)
- DBSecurityGroupQuotaExceededFault (struttura)
- DBSnapshotAlreadyExistsFault (struttura)
- DBSnapshotNotFoundFault (struttura)
- DBSubnetGroupAlreadyExistsFault (struttura)
- DBSubnetGroupDoesNotCoverEnoughAZs (struttura)

- DBSubnetGroupNotAllowedFault (struttura)
- DBSubnetGroupNotFoundFault (struttura)
- DBSubnetGroupQuotaExceededFault (struttura)
- DBSubnetQuotaExceededFault (struttura)
- DBUpgradeDependencyFailureFault (struttura)
- DomainNotFoundFault (struttura)
- EventSubscriptionQuotaExceededFault (struttura)
- GlobalClusterAlreadyExistsFault (struttura)
- GlobalClusterNotFoundFault (struttura)
- GlobalClusterQuotaExceededFault (struttura)
- InstanceQuotaExceededFault (struttura)
- InsufficientDBClusterCapacityFault (struttura)
- InsufficientDBInstanceCapacityFault (struttura)
- InsufficientStorageClusterCapacityFault (struttura)
- InvalidDBClusterEndpointStateFault (struttura)
- InvalidDBClusterSnapshotStateFault (struttura)
- InvalidDBClusterStateFault (struttura)
- InvalidDBInstanceStateFault (struttura)
- InvalidDBParameterGroupStateFault (struttura)
- InvalidDBSecurityGroupStateFault (struttura)
- InvalidDBSnapshotStateFault (struttura)
- InvalidDBSubnetGroupFault (struttura)
- InvalidDBSubnetGroupStateFault (struttura)
- InvalidDBSubnetStateFault (struttura)
- InvalidEventSubscriptionStateFault (struttura)
- InvalidGlobalClusterStateFault (struttura)
- InvalidOptionGroupStateFault (struttura)
- InvalidRestoreFault (struttura)
- InvalidSubnet (struttura)
- InvalidVPCNetworkStateFault (struttura)

- KMSKeyNotAccessibleFault (struttura)
- OptionGroupNotFoundFault (struttura)
- PointInTimeRestoreNotEnabledFault (struttura)
- ProvisionedIopsNotAvailableInAZFault (struttura)
- ResourceNotFoundFault (struttura)
- SNSInvalidTopicFault (struttura)
- SNSNoAuthorizationFault (struttura)
- SNSTopicArnNotFoundFault (struttura)
- SharedSnapshotQuotaExceededFault (struttura)
- SnapshotQuotaExceededFault (struttura)
- SourceNotFoundFault (struttura)
- StorageQuotaExceededFault (struttura)
- StorageTypeNotSupportedFault (struttura)
- SubnetAlreadyInUse (struttura)
- SubscriptionAlreadyExistFault (struttura)
- SubscriptionCategoryNotFoundFault (struttura)
- SubscriptionNotFoundFault (struttura)

AuthorizationAlreadyExistsFault (struttura)

Codice di stato HTTP restituito: 400.

Il CIDRIP o gruppo di sicurezza EC2 specificato è già autorizzato per il gruppo di sicurezza DB specificato.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

AuthorizationNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

Il CIDRIP o gruppo di sicurezza EC2 specificato non è autorizzato per il gruppo di sicurezza DB specificato.

Neptune potrebbe anche non essere autorizzato tramite IAM a eseguire operazioni necessarie per proprio conto.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

AuthorizationQuotaExceededFault (struttura)

Codice di stato HTTP restituito: 400.

La quota di autorizzazione del gruppo di sicurezza DB è stato raggiunto.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

CertificateNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

CertificateIdentifier non si riferisce a un certificato esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBClusterAlreadyExistsFault (struttura)

Codice di stato HTTP restituito: 400.

L'utente ha già un cluster di database con l'identificatore specificato.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBClusterNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

DBClusterIdentifier non si riferisce a un cluster di database esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBClusterParameterGroupNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

DBClusterParameterGroupName non fa riferimento a un gruppo di parametri del cluster di database esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBClusterQuotaExceededFault (struttura)

Codice di stato HTTP restituito: 403.

L'utente ha tentato di creare un nuovo cluster di database e l'utente ha già raggiunto il numero massimo consentito di quota di cluster di database.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

Messaggio che descrive i dettagli del problema.

DBClusterRoleAlreadyExistsFault (struttura)

Codice di stato HTTP restituito: 400.

Il nome della risorsa Amazon (ARN) del ruolo IAM specificato è già associato al cluster database specificato.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBClusterRoleNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

Il nome della risorsa Amazon (ARN) del ruolo IAM specificato non è associato al cluster database specificato.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBClusterRoleQuotaExceededFault (struttura)

Codice di stato HTTP restituito: 400.

È stato superato il numero massimo di ruoli IAM che possono essere associati al cluster di database specificato.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBClusterSnapshotAlreadyExistsFault (struttura)

Codice di stato HTTP restituito: 400.

L'utente ha già uno snapshot di cluster di database con l'identificatore specificato.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBClusterSnapshotNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

DBClusterSnapshotIdentifier non si riferisce a uno snapshot di cluster di database esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBInstanceAlreadyExistsFault (struttura)

Codice di stato HTTP restituito: 400.

L'utente ha già un istanza database con l'identificatore specificato.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBInstanceNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

DBInstanceIdentifier non si riferisce a un'istanza database esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBLogFileNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

LogFileName non si riferisce a un file log di database esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBParameterGroupAlreadyExistsFault (struttura)

Codice di stato HTTP restituito: 400.

Esiste un gruppo di parametri database con lo stesso nome.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBParameterGroupNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

DBParameterGroupName non fa riferimento a un gruppo di parametri database esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBParameterGroupQuotaExceededFault (struttura)

Codice di stato HTTP restituito: 400.

Una richiesta comporterebbe il superamento da parte dell'utente del numero consentito di gruppi di parametri database.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBSecurityGroupAlreadyExistsFault (struttura)

Codice di stato HTTP restituito: 400.

Un gruppo di sicurezza database con il nome specificato in DBSecurityGroupName esiste già.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBSecurityGroupNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

DBSecurityGroupName non fa riferimento a un gruppo di sicurezza DB esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBSecurityGroupNotSupportedFault (struttura)

Codice di stato HTTP restituito: 400.

Un gruppo di sicurezza database non è consentito per questa operazione.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBSecurityGroupQuotaExceededFault (struttura)

Codice di stato HTTP restituito: 400.

Una richiesta comporterebbe il superamento da parte dell'utente del numero consentito di gruppi di sicurezza DB.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBSnapshotAlreadyExistsFault (struttura)

Codice di stato HTTP restituito: 400.

DBSnapshotIdentifier è già utilizzato da uno snapshot esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBSnapshotNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

DBSnapshotIdentifier non si riferisce a uno snapshot DB esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBSubnetGroupAlreadyExistsFault (struttura)

Codice di stato HTTP restituito: 400.

DBSubnetGroupName è già utilizzato da un gruppo di sottoreti DB esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBSubnetGroupDoesNotCoverEnoughAZs (struttura)

Codice di stato HTTP restituito: 400.

Le sottoreti nel gruppo di sottoreti DB dovrebbero coprire almeno due zone di disponibilità a meno che ci sia solo una sola zona di disponibilità.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBSubnetGroupNotAllowedFault (struttura)

Codice di stato HTTP restituito: 400.

Indica che il DBSubnetGroup non deve essere specificato durante la creazione di repliche di lettura che si trovano nella stessa regione dell'istanza di origine.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

Messaggio che descrive i dettagli del problema.

DBSubnetGroupNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

DBSubnetGroupName non fa riferimento a un gruppo di sottoreti DB esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBSubnetGroupQuotaExceededFault (struttura)

Codice di stato HTTP restituito: 400.

Una richiesta comporterebbe il superamento da parte dell'utente del numero consentito di gruppi di sottoreti DB.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBSubnetQuotaExceededFault (struttura)

Codice di stato HTTP restituito: 400.

Una richiesta comporterebbe il superamento da parte dell'utente del numero consentito di sottoreti in un gruppo di sottoreti DB.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DBUpgradeDependencyFailureFault (struttura)

Codice di stato HTTP restituito: 400.

L'aggiornamento del database non riuscito perché non è stato possibile modificare una risorsa dalla quale dipende il DB.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

DomainNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

Dominio non fa riferimento a un dominio Active Directory esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

EventSubscriptionQuotaExceededFault (struttura)

Codice di stato HTTP restituito: 400.

È stato superato il numero di eventi al quale è possibile iscriversi.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

GlobalClusterAlreadyExistsFault (struttura)

Codice di stato HTTP restituito: 400.

`GlobalClusterIdentifier` esiste già. Scegli un nuovo identificatore globale del database (nome univoco) per creare un nuovo cluster database globale.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

GlobalClusterNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

`GlobalClusterIdentifier` non si riferisce a un cluster database globale esistente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

GlobalClusterQuotaExceededFault (struttura)

Codice di stato HTTP restituito: 400.

Il numero di cluster database globali per questo account è già quello massimo consentito.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InstanceQuotaExceededFault (struttura)

Codice di stato HTTP restituito: 400.

Una richiesta comporterebbe il superamento da parte dell'utente del numero consentito di istanze database.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InsufficientDBClusterCapacityFault (struttura)

Codice di stato HTTP restituito: 403.

Il cluster di database non dispone di sufficiente capacità per l'operazione corrente.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InsufficientDBInstanceCapacityFault (struttura)

Codice di stato HTTP restituito: 400.

La classe di istanza database specificata non è disponibile nella zona di disponibilità specificata.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InsufficientStorageClusterCapacityFault (struttura)

Codice di stato HTTP restituito: 400.

Non è disponibile sufficiente storage per l'operazione corrente. È possibile risolvere questo errore aggiornando il gruppo di sottoreti per utilizzare diverse zone di disponibilità che dispongono di maggiore storage.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

Messaggio che descrive i dettagli del problema.

InvalidDBClusterEndpointStateFault (struttura)

Codice di stato HTTP restituito: 400.

L'operazione richiesta non può essere eseguita sull'endpoint mentre si trova in questo stato.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidDBClusterSnapshotStateFault (struttura)

Codice di stato HTTP restituito: 400.

Il valore fornito non è uno stato di snapshot del cluster di database valido.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidDBClusterStateFault (struttura)

Codice di stato HTTP restituito: 400.

Il cluster di database non è in uno stato valido.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidDBInstanceStateFault (struttura)

Codice di stato HTTP restituito: 400.

L'istanza database specificata non è nello stato disponibile.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidDBParameterGroupStateFault (struttura)

Codice di stato HTTP restituito: 400.

Il gruppo di parametri database è in uso o è in uno stato non valido. Se si sta tentando di eliminare il gruppo di parametri, non è possibile eliminarlo quando il gruppo di parametri è in questo stato.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidDBSecurityGroupStateFault (struttura)

Codice di stato HTTP restituito: 400.

Lo stato del gruppo di sicurezza DB non consente l'eliminazione.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidDBSnapshotStateFault (struttura)

Codice di stato HTTP restituito: 400.

Lo stato dello snapshot DB non consente l'eliminazione.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidDBSubnetGroupFault (struttura)

Codice di stato HTTP restituito: 400.

Indica che il DBSubnetGroup non appartiene allo stesso VPC di una replica di lettura esistente tra regioni della stessa istanza di origine.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidDBSubnetGroupStateFault (struttura)

Codice di stato HTTP restituito: 400.

Il gruppo di sottoreti DB non può essere eliminato perché è in uso.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidDBSubnetStateFault (struttura)

Codice di stato HTTP restituito: 400.

La sottorete DB non è nello stato disponibile.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidEventSubscriptionStateFault (struttura)

Codice di stato HTTP restituito: 400.

L'abbonamento a eventi è in uno stato non valido.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidGlobalClusterStateFault (struttura)

Codice di stato HTTP restituito: 400.

Il cluster globale si trova in uno stato non valido e non può eseguire l'operazione richiesta.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidOptionGroupStateFault (struttura)

Codice di stato HTTP restituito: 400.

Il gruppo di opzioni non è nello stato disponibile.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidRestoreFault (struttura)

Codice di stato HTTP restituito: 400.

Impossibile ripristinare da backup vpc a istanze database non vpc.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidSubnet (struttura)

Codice di stato HTTP restituito: 400.

La sottorete richiesta non è valida o più sottoreti sono state richieste che non si trovano tutte in un VPC comune.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

InvalidVPCNetworkStateFault (struttura)

Codice di stato HTTP restituito: 400.

Il gruppo di sottoreti DB non copre tutte le zone di disponibilità dopo la sua creazione in quanto gli utenti cambiano.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

KMSKeyNotAccessibleFault (struttura)

Codice di stato HTTP restituito: 400.

Errore durante l'accesso alla chiave KMS.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

OptionGroupNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

Impossibile trovare il gruppo di opzioni specificato.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

PointInTimeRestoreNotEnabledFault (struttura)

Codice di stato HTTP restituito: 400.

SourceDBInstanceIdentifier si riferisce a un'istanza database dove BackupRetentionPeriod corrisponde a 0.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

ProvisionedIopsNotAvailableInAZFault (struttura)

Codice di stato HTTP restituito: 400.

Provisioned IOPS non disponibile nella zona di disponibilità specificata.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

ResourceNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

L'ID della risorsa specificata non è stato trovato.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

SNSInvalidTopicFault (struttura)

Codice di stato HTTP restituito: 400.

L'argomento SNS non è valido.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

SNSNoAuthorizationFault (struttura)

Codice di stato HTTP restituito: 400.

Non è presente alcuna autorizzazione SNS.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

SNSTopicArnNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

Non è stato possibile trovare l'ARN dell'argomento SNS.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

SharedSnapshotQuotaExceededFault (struttura)

Codice di stato HTTP restituito: 400.

È stato superato il numero massimo di account che è possibile condividere con uno snapshot DB manuale.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

SnapshotQuotaExceededFault (struttura)

Codice di stato HTTP restituito: 400.

Una richiesta comporterebbe il superamento da parte dell'utente del numero consentito di snapshot DB.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

SourceNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

Non è stato possibile trovare l'origine.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

StorageQuotaExceededFault (struttura)

Codice di stato HTTP restituito: 400.

La richiesta comporterebbero il superamento da parte dell'utente della quantità di storage disponibile consentita in tutte le istanze database.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

StorageTypeNotSupportedFault (struttura)

Codice di stato HTTP restituito: 400.

StorageType specificato non può essere associato all'istanza database.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

SubnetAlreadyInUse (struttura)

Codice di stato HTTP restituito: 400.

La sottorete DB è già in uso nella zona di disponibilità.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

SubscriptionAlreadyExistFault (struttura)

Codice di stato HTTP restituito: 400.

Questo abbonamento esiste già.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

SubscriptionCategoryNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

Non è stato possibile trovare la categoria di abbonamento specificata.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

SubscriptionNotFoundFault (struttura)

Codice di stato HTTP restituito: 404.

Impossibile trovare l'abbonamento specificato.

Campi

- message: questo è un ExceptionMessage di tipo `string` (una stringa con codifica UTF-8).

  Messaggio che descrive i dettagli del problema.

Documentazione di riferimento per le API dei dati di Amazon Neptune

Questo capitolo documenta le operazioni delle API dei dati di Neptune che puoi utilizzare per caricare, accedere a e gestire i dati nel grafo Neptune.

L'API dei dati di Neptune supporta gli SDK per il caricamento di dati, l'esecuzione di query, l'acquisizione di informazioni sui dati e l'esecuzione di operazioni di machine learning. Supporta i linguaggi di query Gremlin e openCypher in Neptune ed è disponibile in tutti i linguaggi degli SDK. Firma automaticamente le richieste delle API e semplifica enormemente l'integrazione di Neptune nelle applicazioni.

Indice

- BadRequestException (struttura)
- BulkLoadIdNotFoundException (struttura)
- CancelledByUserException (struttura)
- ClientTimeoutException (struttura)
- ConcurrentModificationException (struttura)
- ConstraintViolationException (struttura)
- ExpiredStreamException (struttura)
- FailureByQueryException (struttura)
- IllegalArgumentException (struttura)
- InternalFailureException (struttura)
- InvalidArgumentException (struttura)
- InvalidNumericDataException (struttura)
- InvalidParameterException (struttura)
- LoadUrlAccessDeniedException (struttura)
- MalformedQueryException (struttura)
- MemoryLimitExceededException (struttura)
- MethodNotAllowedException (struttura)
- MissingParameterException (struttura)
- MLResourceNotFoundException (struttura)
- ParsingException (struttura)
- PreconditionsFailedException (struttura)
- QueryLimitExceededException (struttura)
- QueryLimitException (struttura)
- QueryTooLargeException (struttura)
- ReadOnlyViolationException (struttura)
- S3Exception (struttura)
- ServerShutdownException (struttura)
- StatisticsNotAvailableException (struttura)
- StreamRecordsNotFoundException (struttura)
- ThrottlingException (struttura)

- TimeLimitExceededException (struttura)
- TooManyRequestsException (struttura)
- UnsupportedOperationException (struttura)
- UnloadUrlAccessDeniedException (struttura)

Motore del piano di dati Neptune, ripristino rapido e API della struttura generale

Operazioni del motore:

- GetEngineStatus (azione)
- ExecuteFastReset (azione)

Strutture delle operazioni del motore:

- QueryLanguageVersion (struttura)
- FastResetToken (struttura)

GetEngineStatus (azione)

> Il nome AWS CLI per questa API è: `get-engine-status`.

Recupera lo stato del database a grafo sull'host.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:GetEngineStatus nel cluster.

Richiesta

- Nessun parametro della richiesta.

Risposta

- dbEngineVersion: una stringa di tipo `string` (una stringa con codifica UTF-8).

Imposta la versione del motore Neptune in esecuzione sul cluster database. Se è stata applicata una patch manualmente a questa versione del motore dopo il rilascio, il numero della versione ha il prefisso `Patch-`.

- dfeQueryEngine: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Da impostare su `enabled` se il motore DFE è completamente abilitato o su `viaQueryHint` (impostazione predefinita) se il motore DFE viene utilizzato solo con query per le quali il suggerimento `useDFE` è impostato su `true`.
- features: un array di mappa con coppie chiave-valore in cui:

  Ogni chiave è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Ogni valore è un documento di tipo `document` (un contenuto aperto indipendente dal protocollo rappresentato da un modello di dati simile a JSON).

  Contiene informazioni sullo stato delle funzionalità abilitate nel cluster database.
- gremlin: un oggetto QueryLanguageVersion.

  Contiene informazioni sul linguaggio di query Gremlin disponibile nel cluster. In particolare, contiene un campo version che specifica la versione corrente di TinkerPop utilizzata dal motore.
- labMode: un array di mappa con coppie chiave-valore in cui:

  Ogni chiave è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Ogni valore è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene le impostazioni della modalità Lab utilizzate dal motore.
- opencypher: un oggetto QueryLanguageVersion.

  Contiene informazioni sul linguaggio di query openCypher disponibile nel cluster. In particolare, contiene un campo version che specifica la versione corrente di operCypher utilizzata dal motore.
- role: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Impostato su `reader` se l'istanza è una replica di lettura o su `writer` se è l'istanza principale.
- rollingBackTrxCount: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Se alcune transazioni vengono ripristinate, questo campo contiene il numero di transazioni. Se non ne esistono, il campo non viene visualizzato.

- rollingBackTrxEarliestStartTime: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Impostato sull'ora di inizio della transazione più recente in fase di ripristino. Se non è stato eseguito il rollback di alcuna transazione, il campo non viene visualizzato.

- settings: un array di mappa con coppie chiave-valore in cui:

  Ogni chiave è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Ogni valore è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene informazioni sulle impostazioni correnti sul cluster database. Ad esempio, contiene impostazione attuale per il timeout delle query del cluster (`clusterQueryTimeoutInMs`).

- sparql: un oggetto QueryLanguageVersion.

  Contiene informazioni sul linguaggio di query SPARQL disponibile nel cluster. In particolare, contiene un campo version che specifica la versione corrente di SPARQL utilizzata dal motore.

- startTime: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Impostato sull'ora UTC in cui è iniziato il processo del server corrente.

- status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Impostato su `healthy` se l'istanza non presenta problemi. Se l'istanza è in fase di ripristino dopo un arresto anomalo o un riavvio e ci sono transazioni attive in esecuzione dall'ultimo arresto del server, lo stato è impostato su `recovery`.

Errori

- UnsupportedOperationException
- InternalFailureException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- IllegalArgumentException
- TooManyRequestsException

ExecuteFastReset (azione)

Il nome AWS CLI per questa API è: `execute-fast-reset`.

La REST API di ripristino rapido consente di ripristinare un grafo Neptune in modo veloce e semplice, rimuovendo tutti i relativi dati.

La procedura di ripristino rapido di Neptune prevede due fasi. Per prima cosa, occorre chiamare `ExecuteFastReset` con `action` impostato su `initiateDatabaseReset`. Viene così restituito un token UUID da includere quando si chiama di nuovo `ExecuteFastReset` con `action` impostato su `performDatabaseReset`. Consulta Svuotamento di un cluster database Amazon Neptune utilizzando l'API di ripristino rapido.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:ResetDatabase nel cluster.

Richiesta

- action (nella CLI: `--action`): Obbligatorio: un'azione di tipo `string` (una stringa con codifica UTF-8).

  L'azione di ripristino rapido. Uno dei seguenti valori:

  - **`initiateDatabaseReset`**: questa azione genera un token unico necessario per eseguire effettivamente il ripristino rapido.
  - **`performDatabaseReset`**: questa azione utilizza il token generato dall'azione `initiateDatabaseReset` per eseguire effettivamente il ripristino rapido.

- token (nella CLI: `--token`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il token per avviare il ripristino rapido.

Risposta

- payload: un oggetto FastResetToken.

  Il parametro `payload` viene restituito solo dall'azione `initiateDatabaseReset` e contiene il token univoco da utilizzare con l'azione `performDatabaseReset` per eseguire il ripristino.
- status: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

Il parametro `status` viene restituito solo per l'azione `performDatabaseReset` e indica se la richiesta di ripristino rapido è accettata o meno.

Errori

- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- ServerShutdownException
- PreconditionsFailedException
- MethodNotAllowedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Strutture delle operazioni del motore:

QueryLanguageVersion (struttura)

Struttura per esprimere la versione del linguaggio di query.

Campi

- version: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La versione del linguaggio di query.

FastResetToken (struttura)

Una struttura contenente il token utilizzato per avviare un ripristino rapido.

Campi

- token: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un UUID generato dal database durante l'azione `initiateDatabaseReset` e poi utilizzato da `performDatabaseReset` per reimpostare il database.

API delle query Neptune

Azioni di query Gremlin:

- ExecuteGremlinQuery (azione)
- ExecuteGremlinExplainQuery (azione)
- ExecuteGremlinProfileQuery (azione)
- ListGremlinQueries (azione)
- GetGremlinQueryStatus (azione)
- CancelGremlinQuery (azione)

Azioni di query openCypher:

- ExecuteOpenCypherQuery (azione)
- ExecuteOpenCypherExplainQuery (azione)
- ListOpenCypherQueries (azione)
- GetOpenCypherQueryStatus (azione)
- CancelOpenCypherQuery (azione)

Strutture di query:

- QueryEvalStats (struttura)
- GremlinQueryStatus (struttura)
- GremlinQueryStatusAttributes (struttura)

ExecuteGremlinQuery (azione)

Il nome AWS CLI per questa API è: `execute-gremlin-query`.

Questo comando esegue una query Gremlin. Amazon Neptune è compatibile con Apache TinkerPop3 e Gremlin, quindi puoi usare il linguaggio di attraversamento Gremlin per eseguire query sul il grafo, come descritto nella sezione dedicata al grafo nella documentazione di Apache TinkerPop3. Ulteriori dettagli sono disponibili anche in Accesso a un grafo Neptune con Gremlin.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta una delle seguenti azioni IAM nel cluster, a seconda della query:

- neptune-db:ReadDataViaQuery
- neptune-db:WriteDataViaQuery
- neptune-db:DeleteDataViaQuery

Tieni presente che la chiave di condizione IAM neptune-db:QueryLanguage:Gremlin può essere utilizzata nel documento di policy per limitare l'uso delle query Gremlin (vedi Chiavi di condizioni disponibili nelle dichiarazioni delle policy di accesso ai dati IAM di Neptune).

Richiesta

- gremlinQuery (nella CLI: `--gremlin-query`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Utilizza questa API per eseguire le query Gremlin in formato stringa, in modo simile all'uso dell'endpoint HTTP. L'interfaccia è compatibile con qualsiasi versione di Gremlin utilizzata dal cluster database (consulta la sezione relativa al client Tinkerpop per determinare quali release di Gremlin sono supportate dalla versione del motore).
- serializer (nella CLI: `--serializer`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se diverso da null, i risultati della query vengono restituiti in un messaggio di risposta serializzato nel formato specificato da questo parametro. Vedi la sezione GraphSON nella documentazione di TinkerPop per un elenco dei formati attualmente supportati.

Risposta

- meta: un documento di tipo `document` (un contenuto aperto indipendente dal protocollo rappresentato da un modello di dati simile a JSON).

  I metadati relativi alla query Gremlin.

- requestId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco della query Gremlin.

- result: un documento di tipo `document` (un contenuto aperto indipendente dal protocollo rappresentato da un modello di dati simile a JSON).

  L'output della query Gremlin dal server.

- status: un oggetto GremlinQueryStatusAttributes.

  Lo stato della query Gremlin.

Errori

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException

- ConcurrentModificationException

ExecuteGremlinExplainQuery (azione)

Il nome AWS CLI per questa API è: `execute-gremlin-explain-query`.

Esegue una query "explain" di Gremlin.

Amazon Neptune ha aggiunto una funzionalità Gremlin denominata `explain` che fornisce uno strumento self-service per comprendere l'approccio di esecuzione adottato dal motore Neptune per la query. È possibile richiamarla aggiungendo un parametro `explain` a una chiamata HTTP che invia una query Gremlin.

La funzionalità "explain" fornisce informazioni sulla struttura logica dei piani di esecuzione delle query. Puoi utilizzare queste informazioni per identificare potenziali problemi di valutazione ed esecuzione, oltre che per ottimizzare la query, come spiegato in Ottimizzazione delle query Gremlin. Puoi quindi utilizzare gli hint di query per migliorare i piani di esecuzione delle query.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta una delle seguenti azioni IAM nel cluster, a seconda della query:

- neptune-db:ReadDataViaQuery
- neptune-db:WriteDataViaQuery
- neptune-db:DeleteDataViaQuery

Tieni presente che la chiave di condizione IAM neptune-db:QueryLanguage:Gremlin può essere utilizzata nel documento di policy per limitare l'uso delle query Gremlin (vedi Chiavi di condizioni disponibili nelle dichiarazioni delle policy di accesso ai dati IAM di Neptune).

Richiesta

- gremlinQuery (nella CLI: `--gremlin-query`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La stringa della query "explain" di Gremlin.

Risposta

- output: un valore ReportAsText di tipo `blob` (un blocco di dati binari non interpretati).

  Un blob di testo contenente il risultato della query "explain" di Gremlin, come descritto in Ottimizzazione delle query Gremlin.

Errori

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteGremlinProfileQuery (azione)

Il nome AWS CLI per questa API è: `execute-gremlin-profile-query`.

Esegue una query "profile" di Gremlin che esegue un attraversamento Gremlin specificato, raccoglie varie metriche relative all'esecuzione e produce un report del profilo come output. Per i dettagli, consulta API profile di Gremlin in Neptune.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:ReadDataViaQuery nel cluster.

Tieni presente che la chiave di condizione IAM neptune-db:QueryLanguage:Gremlin può essere utilizzata nel documento di policy per limitare l'uso delle query Gremlin (vedi Chiavi di condizioni disponibili nelle dichiarazioni delle policy di accesso ai dati IAM di Neptune).

Richiesta

- chop (nella CLI: `--chop`): un valore Integer di tipo `integer` (numero intero a 32 bit con segno).

  Se diverso da zero, la stringa dei risultati viene troncata a tale numero di caratteri. Se impostato su zero, la stringa contiene tutti i risultati.
- gremlinQuery (nella CLI: `--gremlin-query`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La stringa di query Gremlin per il profilo.
- indexOps (nella CLI: `--index-ops`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Se il flag è impostato su `TRUE`, mostra un report dettagliato di tutte le operazioni sugli indici eseguite durante l'esecuzione e la serializzazione delle query.
- results (nella CLI: `--results`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Se il flag è impostato su `TRUE`, i risultati della query vengono raccolti e visualizzati come parte del report del profilo. Se `FALSE`, viene visualizzato solo il conteggio dei risultati.
- serializer (nella CLI: `--serializer`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se diverso da null, i risultati raccolti vengono restituiti in un messaggio di risposta serializzato nel formato specificato da questo parametro. Per ulteriori informazioni, consulta API profile di Gremlin in Neptune.

Risposta

- output: un valore ReportAsText di tipo `blob` (un blocco di dati binari non interpretati).

  Un blob di testo contenente il risultato della query profile di Gremlin. Per i dettagli, consulta API profile di Gremlin in Neptune.

Errori

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ListGremlinQueries (azione)

Il nome AWS CLI per questa API è: `list-gremlin-queries`.

Elenca le query Gremlin attive. Per i dettagli sull'output, vedi API di stato delle query Gremlin.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:GetQueryStatus nel cluster.

Tieni presente che la chiave di condizione IAM neptune-db:QueryLanguage:Gremlin può essere utilizzata nel documento di policy per limitare l'uso delle query Gremlin (vedi Chiavi di condizioni disponibili nelle dichiarazioni delle policy di accesso ai dati IAM di Neptune).

Richiesta

- includeWaiting (nella CLI: `--include-waiting`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `TRUE`, l'elenco restituito include le query in attesa. Il valore predefinito è `FALSE`;

Risposta

- acceptedQueryCount: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Numero di query accettate ma non ancora completate, incluse le query nella coda.
- queries: una matrice di oggetti GremlinQueryStatus.

  Un elenco delle query correnti.
- runningQueryCount: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Il numero di query Gremlin attualmente in esecuzione.

Errori

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException

- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

GetGremlinQueryStatus (azione)

Il nome AWS CLI per questa API è: `get-gremlin-query-status`.

Ottiene lo stato di una query Gremlin specificata.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:GetQueryStatus nel cluster.

Tieni presente che la chiave di condizione IAM neptune-db:QueryLanguage:Gremlin può essere utilizzata nel documento di policy per limitare l'uso delle query Gremlin (vedi Chiavi di condizioni disponibili nelle dichiarazioni delle policy di accesso ai dati IAM di Neptune).

Richiesta

- queryId (nella CLI: `--query-id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco della query Gremlin.

Risposta

- queryEvalStats: un oggetto QueryEvalStats.

  Lo stato di valutazione della query Gremlin.
- queryId: una stringa di tipo `string` (una stringa con codifica UTF-8).

L'ID della query per la quale viene restituito lo stato.

- queryString: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La stringa di query Gremlin.

Errori

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

CancelGremlinQuery (azione)

Il nome AWS CLI per questa API è: `cancel-gremlin-query`.

Annulla una query Gremlin. Per ulteriori informazioni, consulta Annullamento delle query Gremlin.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:CancelQuery nel cluster.

Richiesta

- queryId (nella CLI: `--query-id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco della query da annullare.

Risposta

- status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato dell'annullamento

Errori

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

Azioni di query openCypher:

ExecuteOpenCypherQuery (azione)

Il nome AWS CLI per questa API è: `execute-open-cypher-query`.

Esegue una query openCypher. Per ulteriori informazioni, consulta Accesso al grafo Neptune con openCypher.

Neptune supporta la creazione di applicazioni a grafo mediante openCypher, che è attualmente uno dei linguaggi di query più popolari tra gli sviluppatori che lavorano con database a grafo. Sviluppatori, analisti aziendali e data scientist apprezzano la sintassi dichiarativa di openCypher ispirata a SQL perché fornisce una struttura familiare mediante la quale eseguire query su grafi di proprietà.

Il linguaggio openCypher è stato sviluppato originariamente da Neo4j, per poi diventare open source nel 2015, e ha contribuito al progetto openCypher con una licenza open source Apache 2.

Tieni presente che, quando richiami questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta una delle seguenti azioni IAM nel cluster, a seconda della query:

- neptune-db:ReadDataViaQuery
- neptune-db:WriteDataViaQuery
- neptune-db:DeleteDataViaQuery

Tieni inoltre presente che la chiave di condizione IAM neptune-db:QueryLanguage:OpenCypher può essere utilizzata nel documento di policy per limitare l'uso delle query openCypher (vedi Chiavi di condizioni disponibili nelle dichiarazioni delle policy di accesso ai dati IAM di Neptune).

Richiesta

- openCypherQuery (nella CLI: `--open-cypher-query`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La stringa di query openCypher da eseguire.
- parameters (nella CLI: `--parameters`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  I parametri di query openCypher per l'esecuzione delle query. Per ulteriori informazioni, consulta Esempi di query con parametri openCypher.

Risposta

- results: Obbligatorio: un documento di tipo `document` (un contenuto aperto indipendente dal protocollo rappresentato da un modello di dati simile a JSON).

I risultati della query openCypher.

Errori

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteOpenCypherExplainQuery (azione)

Il nome AWS CLI per questa API è: `execute-open-cypher-explain-query`.

Esegue una richiesta `explain` openCypher. Per ulteriori informazioni, consulta Funzione explain di openCypher.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:ReadDataViaQuery nel cluster.

Tieni presente che la chiave di condizione IAM neptune-db:QueryLanguage:OpenCypher può essere utilizzata nel documento di policy per limitare l'uso delle query openCypher (vedi Chiavi di condizioni disponibili nelle dichiarazioni delle policy di accesso ai dati IAM di Neptune).

Richiesta

- explainMode (nella CLI: `--explain-mode`): Obbligatorio: un valore OpenCypherExplainMode di tipo `string` (una stringa con codifica UTF-8).

  La modalità `explain` di openCypher. Può essere `static`, `dynamic` o `details`.
- openCypherQuery (nella CLI: `--open-cypher-query`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La stringa di query openCypher.
- parameters (nella CLI: `--parameters`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  I parametri della query openCypher.

Risposta

- results: Obbligatorio: un blob di tipo `blob` (un blocco di dati binari non interpretati).

  Un blob di testo contenente i risultati della query `explain` di openCypher.

Errori

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException

- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ListOpenCypherQueries (azione)

Il nome AWS CLI per questa API è: `list-open-cypher-queries`.

Elenca le query openCypher attive. Per ulteriori informazioni, consulta Endpoint dello stato openCypher di Neptune.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:GetQueryStatus nel cluster.

Tieni presente che la chiave di condizione IAM neptune-db:QueryLanguage:OpenCypher può essere utilizzata nel documento di policy per limitare l'uso delle query openCypher (vedi Chiavi di condizioni disponibili nelle dichiarazioni delle policy di accesso ai dati IAM di Neptune).

Richiesta

- includeWaiting (nella CLI: `--include-waiting`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

Se impostato su TRUE e non sono presenti altri parametri, vengono restituite le informazioni sullo stato per le query in attesa e per quelle in esecuzione.

Risposta

- acceptedQueryCount: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Numero di query accettate ma non ancora completate, incluse le query nella coda.
- queries: una matrice di oggetti GremlinQueryStatus.

  Un elenco delle query openCypher correnti.
- runningQueryCount: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Il numero di query openCypher attualmente in esecuzione.

Errori

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException

- ConcurrentModificationException

GetOpenCypherQueryStatus (azione)

Il nome AWS CLI per questa API è: `get-open-cypher-query-status`.

Recupera lo stato di una query openCypher specificata.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:GetQueryStatus nel cluster.

Tieni presente che la chiave di condizione IAM neptune-db:QueryLanguage:OpenCypher può essere utilizzata nel documento di policy per limitare l'uso delle query openCypher (vedi Chiavi di condizioni disponibili nelle dichiarazioni delle policy di accesso ai dati IAM di Neptune).

Richiesta

- queryId (nella CLI: `--query-id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID univoco della query openCypher per la quale recuperare lo stato.

Risposta

- queryEvalStats: un oggetto QueryEvalStats.

  Lo stato di valutazione della query openCypher.
- queryId: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID univoco della query per la quale viene restituito lo stato.
- queryString: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La stringa di query openCypher.

Errori

- InvalidNumericDataException
- BadRequestException

- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

CancelOpenCypherQuery (azione)

Il nome AWS CLI per questa API è: `cancel-open-cypher-query`.

Annulla una query openCypher specificata. Per ulteriori informazioni, consulta Endpoint dello stato openCypher di Neptune.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:CancelQuery nel cluster.

Richiesta

- queryId (nella CLI: `--query-id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID univoco della query openCypher da annullare.
- silent (nella CLI: `--silent`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

Se impostato su `TRUE`, l'annullamento della query openCypher avviene senza alcun avviso.

Risposta

- payload: un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Il payload dell'annullamento per la query openCypher.
- status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato di annullamento della query openCypher.

Errori

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

Strutture di query:

QueryEvalStats (struttura)

Struttura per acquisire statistiche sulle query, ad esempio quante query sono in esecuzione, accettate o in attesa e i relativi dettagli.

Campi

- cancelled: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Impostato su `TRUE` se la query è stata annullata, in caso contrario FALSE.
- elapsed: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

  Il numero di millisecondi durante i quali è stata eseguita la query (fino a questo momento).
- subqueries: questo è un documento di tipo `document` (un contenuto aperto indipendente dal protocollo rappresentato da un modello di dati simile a JSON).

  Numero di sottoquery in questa query.
- waited: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

  Indica il tempo di attesa della query, in millisecondi.

GremlinQueryStatus (struttura)

Acquisisce lo stato di una query Gremlin (vedi la pagina API di stato delle query Gremlin).

Campi

- queryEvalStats: questo è un oggetto QueryEvalStats.

  Le statistiche della query Gremlin.
- queryId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della query Gremlin.
- queryString: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  La stringa della query Gremlin.

GremlinQueryStatusAttributes (struttura)

Contiene i componenti di stato di una query Gremlin.

Campi

- attributes: questo è un documento di tipo `document` (un contenuto aperto indipendente dal protocollo rappresentato da un modello di dati simile a JSON).

  Attributi dello stato della query Gremlin.
- code: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

  Il codice di risposta HTTP restituito dalla richiesta di query Gremlin.
- message: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio sullo stato.

API dello strumento di caricamento in blocco per il piano dati Neptune

Azioni di caricamento in blocco:

- StartLoaderJob (azione)
- GetLoaderJobStatus (azione)
- ListLoaderJobs (azione)
- CancelLoaderJob (azione)

Struttura del caricamento in blocco:

- LoaderIdResult (struttura)

StartLoaderJob (azione)

Il nome AWS CLI per questa API è: `start-loader-job`.

Avvia un processo dello strumento di caricamento in blocco Neptune per caricare i dati da un bucket Amazon S3 in un'istanza database Neptune. Vedi Uso dello strumento di caricamento in blocco Amazon Neptune per importare i dati.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:StartLoaderJob nel cluster.

Richiesta

- dependencies (nella CLI: `--dependencies`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Si tratta di un parametro facoltativo che può rendere subordinata una richiesta di caricamento in coda al completamento di una o più attività precedenti nella coda.

  Neptune può accodare fino a 64 richieste di caricamento alla volta, se i relativi parametri `queueRequest` sono impostati su `"TRUE"`. Il parametro `dependencies` consente di rendere l'esecuzione di tale richiesta in coda dipendente dal completamento corretto di una o più richieste precedenti specificate nella coda.

  Ad esempio, se `Job-A` e `Job-B` del caricamento sono indipendenti l'una dall'altra, ma `Job-C` richiede che `Job-A` e `Job-B` siano completate prima del suo avvio, procedere come segue:

  1. Inviare `load-job-A` e `load-job-B` una dopo l'altra in qualsiasi ordine e salvare i loro id di caricamento.
  2. Inviare `load-job-C` con gli id di caricamento delle due attività nel campo `dependencies`:

  Example

  ```
  "dependencies" : ["(job_A_load_id)", "(job_B_load_id)"]
  ```

  A causa del parametro `dependencies`, lo strumento di caricamento in blocco non avvia `Job-C` fino a quando `Job-A` e `Job-B` non sono state completate correttamente. Se una di queste attività non riesce, l'attività non verrà eseguita e il suo stato sarà impostato su `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

  È possibile impostare più livelli di dipendenza in questo modo, in modo che l'errore di un'attività causi l'annullamento di tutte le richieste direttamente o indirettamente dipendenti da essa.

- failOnError (nella CLI: `--fail-on-error`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  `failOnError`: un flag per attivare un arresto completo in caso di errore.

  Valori consentiti: `"TRUE"`, `"FALSE"`.

  Valore predefinito: `"TRUE"`.

  Quando questo parametro è impostato su `"FALSE"`, lo strumento di caricamento tenta di caricare tutti i dati nella posizione specificata, saltando eventuali voci con errori.

  Quando questo parametro è impostato su `"TRUE"`, lo dello strumento di caricamento si arresta non appena rileva un errore. I dati caricati fino a quel punto persistono.
- format (nella CLI: `--format`): Obbligatorio: un formato di tipo `string` (una stringa con codifica UTF-8).

  Il formato dei dati. Per ulteriori informazioni sui formati di dati per il comando `Loader` di Neptune, consulta Formati dei dati da caricare.

  Valori consentiti

  - **`csv`** per il formato dei dati CSV Gremlin.
  - **`opencypher`** per il formato dei dati CSV openCypher.
  - **`ntriples`** per il formato dei dati N-Triples RDF.
  - **`nquads`** per il formato dei dati N-Quads RDF.
  - **`rdfxml`** per il formato dei dati RDF\XML RDF.
  - **`turtle`** per il formato dei dati Turtle RDF.
- iamRoleArn (nella CLI: `--iam-role-arn`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della risorsa Amazon (ARN) per un ruolo IAM che deve essere assunto dall'istanza database Neptune per l'accesso al bucket S3. L'ARN del ruolo IAM fornito qui deve essere collegato al cluster database (vedi Aggiunta del ruolo IAM a un cluster Amazon Neptune).
- mode (nella CLI: `--mode`): una modalità di tipo `string` (una stringa con codifica UTF-8).

  Modalità del processo di caricamento

  Valori consentiti: `RESUME`, `NEW`, `AUTO`.

Valore predefinito: `AUTO`.

- `RESUME`: in modalità RESUME, lo strumento di caricamento cerca un caricamento precedente da questa origine e, se ne trova uno, riprende l'attività di caricamento. Se non viene trovata alcuna attività di caricamento precedente, lo strumento di caricamento si arresta.

  Lo strumento di caricamento evita di ricaricare i file caricati correttamente in un'attività precedente. Tenta di elaborare solo i file non caricati. Se sono stati eliminati i dati caricati in precedenza dal cluster Neptune, tali dati non vengono ricaricati in questa modalità. Se un processo di caricamento precedente ha caricato correttamente tutti i file dalla stessa origine, nulla viene ricaricato e lo strumento di caricamento restituisce un risultato positivo.
- `NEW`: in modalità NEW viene creata una nuova richiesta di caricamento, indipendentemente da eventuali caricamenti precedenti. Questa modalità può essere utilizzata per ricaricare tutti i dati provenienti da un'origine dopo che sono stati eliminati dati caricati precedentemente dal cluster Neptune o per caricare nuovi dati disponibili nella stessa origine.
- `AUTO`: in modalità AUTO, lo strumento di caricamento cerca un'attività di caricamento precedente dalla stessa origine e, se ne trova una, riprende tale attività, proprio come in modalità `RESUME`.

  Se lo strumento di caricamento non trova un'attività di caricamento precedente dalla stessa origine, carica tutti i dati dall'origine, proprio come in modalità `NEW`.

- parallelism (nella CLI: `--parallelism`): un parallelismo di tipo `string` (una stringa con codifica UTF-8).

  Il parametro facoltativo `parallelism` può essere impostato per ridurre il numero di thread utilizzati dall'attività di caricamento in blocco.

  Valori consentiti:

  - `LOW`: il numero di thread utilizzati è il numero di vCPU disponibili diviso per 8.
  - `MEDIUM`: il numero di thread utilizzati è il numero di vCPU disponibili diviso per 2.
  - `HIGH`: il numero di thread utilizzati corrisponde al numero di vCPU disponibili.
  - `OVERSUBSCRIBE`: il numero di thread utilizzati è il numero di vCPU disponibili moltiplicato per 2. Se viene utilizzato questo valore, lo strumento di caricamento in blocco occupa tutte le risorse disponibili.

Ciò non significa, tuttavia, che l'impostazione `OVERSUBSCRIBE` comporti un utilizzo della CPU al 100%. Poiché l'operazione di caricamento è limitata dalle attività di I/O, l'utilizzo massimo della CPU previsto è compreso tra il 60% e il 70%.

Valore predefinito: `HIGH`

L'impostazione `parallelism` a volte può causare un deadlock tra i thread durante il caricamento dei dati openCypher. Quando ciò accade, Neptune restituisce l'errore `LOAD_DATA_DEADLOCK`. Di solito è possibile risolvere il problema impostando `parallelism` su un valore inferiore e riprovando il comando di caricamento.

- parserConfiguration (nella CLI: `--parser-configuration`): un array di mappa con coppie chiave-valore in cui:

  Ogni chiave è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Ogni valore è una stringa di tipo `string` (una stringa con codifica UTF-8).

  `parserConfiguration`: un oggetto opzionale con valori di configurazione del parser aggiuntivi. Ciascuno dei parametri figlio è anche facoltativo:

  - **`namedGraphUri`**: il grafo predefinito per tutti i formati RDF quando non viene specificato alcun grafo (per formati non quads e voci NQUAD senza grafo).

    Il valore predefinito è `https://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

  - **`baseUri`**: l'URI di base per i formati RDF/XML e Turtle.

    Il valore predefinito è `https://aws.amazon.com/neptune/default`.

  - **`allowEmptyStrings`**: gli utenti di Gremlin devono essere in grado di passare valori di stringa vuoti ("") come proprietà dei nodi e degli archi durante il caricamento di dati CSV. Se `allowEmptyStrings` è impostato su `false` (valore predefinito), le stringhe vuote vengono trattate come valori null e non vengono caricate.

    Se `allowEmptyStrings` è impostato su `true`, lo strumento di caricamento considera le stringhe vuote come valori di proprietà validi e le carica di conseguenza.

- queueRequest (nella CLI: `--queue-request`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

Si tratta di un parametro flag opzionale che indica se la richiesta di caricamento può essere accodata o meno.

Non è necessario attendere il completamento di un processo di caricamento prima di emettere quello successivo, perché Neptune può accodare fino a 64 processi alla volta, a condizione che i relativi parametri `queueRequest` siano tutti impostati su `"TRUE"`. L'ordine di accodamento dei processi sarà first-in-first-out (FIFO).

Se il parametro `queueRequest` viene omesso o impostato su `"FALSE"`, la richiesta di caricamento avrà esito negativo se un'altra attività di caricamento è già in esecuzione.

Valori consentiti: `"TRUE"`, `"FALSE"`.

Valore predefinito: `"FALSE"`.

- s3BucketRegion (nella CLI: `--s-3-bucket-region`): Obbligatorio: un valore S3BucketRegion di tipo `string` (una stringa con codifica UTF-8).

  La regione Amazon del bucket S3. Deve corrispondere alla regione Amazon del cluster database.

- source (nella CLI: `--source`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il parametro `source` accetta un URI S3 che identifica un singolo file, più file, una cartella o più cartelle. Neptune carica ogni file di dati in qualsiasi cartella specificata.

  l'URI può essere in uno dei seguenti formati.

  - `s3://(bucket_name)/(object-key-name)`
  - `https://s3.amazonaws.com/(bucket_name)/(object-key-name)`
  - `https://s3.us-east-1.amazonaws.com/(bucket_name)/(object-key-name)`

  L'elemento `object-key-name` dell'URI è equivalente al parametro prefix in una chiamata API ListObjects di S3. Identifica tutti gli oggetti nel bucket S3 specificato i cui nomi iniziano con il prefisso specificato. Può trattarsi di un singolo file o una singola cartella oppure di più file e/o cartelle.

  La cartella o le cartelle specificate possono contenere più file di vertici e più file di archi.

- updateSingleCardinalityProperties (nella CLI: `--update-single-cardinality-properties`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

`updateSingleCardinalityProperties` è un parametro facoltativo che controlla il modo in cui lo strumento di caricamento in blocco tratta un nuovo valore per le proprietà di vertici o archi a cardinalità singola. Non è supportato per il caricamento di dati openCypher.

Valori consentiti: `"TRUE"`, `"FALSE"`.

Valore predefinito: `"FALSE"`.

Come impostazione predefinita o quando `updateSingleCardinalityProperties` è impostato esplicitamente su `"FALSE"`, lo strumento di caricamento considera un nuovo valore come un errore, perché viola la cardinalità singola.

Quando `updateSingleCardinalityProperties` è impostato invece su `"TRUE"`, lo strumento di caricamento in blocco sostituisce il valore esistente con quello nuovo. Se valori di proprietà multipli edge o vertice a cardinalità singola vengono forniti nei file di origine caricati, il valore finale alla fine del caricamento in blocco potrebbe essere uno qualsiasi di questi nuovi valori. Lo strumento di caricamento garantisce solo che il valore esistente è stato sostituito da uno di quelli nuovi.

- userProvidedEdgeIds (nella CLI: `--user-provided-edge-ids`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Questo parametro è richiesto solo quando si caricano dati openCypher che contengono ID di relazione. Deve essere incluso e impostato su `True` se gli ID delle relazioni openCypher vengono forniti esplicitamente nei dati da caricare (consigliato).

  Se `userProvidedEdgeIds` è assente o è impostato su `True`, in ogni file delle relazioni all'interno del caricamento deve esistere una colonna `:ID`.

  Se `userProvidedEdgeIds` è presente ed è impostato su `False`, i file delle relazioni all'interno del caricamento non devono contenere una colonna `:ID`. Lo strumento di caricamento Neptune genera automaticamente un ID per ogni relazione.

  È utile fornire in modo esplicito gli ID delle relazioni in modo che lo strumento di caricamento possa riprendere il caricamento dopo la correzione dell'errore nei dati CSV, senza dover ricaricare le relazioni già caricate. Se gli ID delle relazioni non sono stati assegnati in modo esplicito, lo strumento di caricamento non può riprendere un caricamento non riuscito se è stato necessario correggere un file delle relazioni, perciò dovrà ricaricare tutte le relazioni.

Risposta

- payload: Obbligatorio: un array di mappa con coppie chiave-valore in cui:

  Ogni chiave è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Ogni valore è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Contiene una coppia nome-valore `loadId` che fornisce un identificatore per l'operazione di caricamento.
- status: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice HTTP restituito che indica lo stato del processo di caricamento.

Errori

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- S3Exception
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

GetLoaderJobStatus (azione)

Il nome AWS CLI per questa API è: `get-loader-job-status`.

Ottiene informazioni su un determinato processo di caricamento. Neptune tiene traccia dei 1.024 lavori di caricamento in blocco più recenti e archivia solo gli ultimi 10.000 dettagli di errore per processo.

Per ulteriori informazioni, consulta API per il recupero dello stato dello strumento di caricamento Neptune.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:GetLoaderJobStatus nel cluster.

Richiesta

- details (nella CLI: `--details`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Flag che indica se includere o meno i dettagli oltre allo stato generale (`TRUE` o `FALSE`, l'impostazione predefinita è `FALSE`).
- errors (nella CLI: `--errors`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Flag che indica se includere o meno un elenco di errori riscontrati (`TRUE` o `FALSE`, l'impostazione predefinita è `FALSE`).

  L'elenco degli errori è paginato. I parametri `page` ed `errorsPerPage` consentono di esaminare tutti gli errori.
- errorsPerPage (nella CLI: `--errors-per-page`): un valore PositiveInteger di tipo `integer` (numero intero a 32 bit con segno), almeno 1 ?st?.

  Il numero di errori restituiti in ogni pagina (un numero intero positivo; l'impostazione predefinita è `10`). È valido solo se il parametro `errors` è impostato su `TRUE`.
- loadId (nella CLI: `--load-id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID del processo di caricamento di cui ottenere lo stato.
- page (nella CLI: `--page`): un valore PositiveInteger di tipo `integer` (numero intero a 32 bit con segno), almeno 1 ?st?.

Il numero della pagina di errore (un numero intero positivo; l'impostazione predefinita è 1). È valido solo se il parametro `errors` è impostato su TRUE.

Risposta

- payload: Obbligatorio: un documento di tipo `document` (un contenuto aperto indipendente dal protocollo rappresentato da un modello di dati simile a JSON).

  Informazioni sullo stato del processo di caricamento, con un layout simile al seguente:

  Example

```
{
          "status" : "200 OK",
          "payload" : {
            "feedCount" : [
              {
                "LOAD_FAILED" : (number)
              }
            ],
            "overallStatus" : {
              "fullUri" : "s3://(bucket)/(key)",
              "runNumber" : (number),
              "retryNumber" : (number),
              "status" : "(string)",
              "totalTimeSpent" : (number),
              "startTime" : (number),
              "totalRecords" : (number),
              "totalDuplicates" : (number),
              "parsingErrors" : (number),
              "datatypeMismatchErrors" : (number),
              "insertErrors" : (number),
            },
            "failedFeeds" : [
              {
                "fullUri" : "s3://(bucket)/(key)",
                "runNumber" : (number),
                "retryNumber" : (number),
                "status" : "(string)",
                "totalTimeSpent" : (number),
                "startTime" : (number),
```

```
                "totalRecords" : (number),
                "totalDuplicates" : (number),
                "parsingErrors" : (number),
                "datatypeMismatchErrors" : (number),
                "insertErrors" : (number),
              }
            ],
            "errors" : {
              "startIndex" : (number),
              "endIndex" : (number),
              "loadId" : "(string),
              "errorLogs" : [ ]
            }
          }
        }
```

- status: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di risposta HTTP per la richiesta.

Errori

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ListLoaderJobs (azione)

Il nome AWS CLI per questa API è: `list-loader-jobs`.

Recupera un elenco di `loadIds` per tutti i processi dello strumento di caricamento attivi.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:ListLoaderJobs nel cluster.

Richiesta

- includeQueuedLoads (nella CLI: `--include-queued-loads`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Un parametro facoltativo che può essere utilizzato per escludere gli ID di caricamento delle richieste di caricamento in coda quando viene richiesto un elenco di ID di caricamento mediante l'impostazione del parametro su `FALSE`. Il valore predefinito è `TRUE`.
- limit (nella CLI: `--limit`): un valore ListLoaderJobsInputLimitInteger di tipo `integer` (numero intero a 32 bit con segno), non meno di 1 o più di 100 ?st?s.

  Il numero di ID di caricamento da elencare. Deve essere un numero intero positivo maggiore di zero e non superiore a `100` (valore predefinito).

Risposta

- payload: obbligatorio: un oggetto LoaderIdResult.

  L'elenco richiesto di ID dei processi.
- status: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Restituisce lo stato della richiesta di elenco dei processi.

Errori

- UnsupportedOperationException
- BadRequestException
- InvalidParameterException

- BulkLoadIdNotFoundException
- InternalFailureException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException

CancelLoaderJob (azione)

Il nome AWS CLI per questa API è: `cancel-loader-job`.

Annulla un processo di caricamento specificato. Questa è una richiesta HTTP `DELETE`. Per ulteriori informazioni, consulta API per il recupero dello stato dello strumento di caricamento Neptune.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:CancelLoaderJob nel cluster.

Richiesta

- loadId (nella CLI: `--load-id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID del processo di caricamento da eliminare.

Risposta

- status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato di annullamento.

Errori

- BadRequestException

- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Struttura del caricamento in blocco:

LoaderIdResult (struttura)

Contiene un elenco di ID di caricamento.

Campi

- loadIds: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di ID di caricamento.

API del piano dati di Neptune Streams

Azioni di accesso ai flussi:

- GetPropertygraphStream (azione)

Strutture di dati dei flussi:

- PropertygraphRecord (struttura)

- PropertygraphData (struttura)

GetPropertygraphStream (azione)

Il nome AWS CLI per questa API è: `get-propertygraph-stream`.

Ottiene un flusso per un grafo delle proprietà.

La funzionalità Neptune Streams consente di generare una sequenza completa di voci di log delle modifiche che registrano automaticamente e in tempo reale ogni modifica apportata ai dati del grafo. `GetPropertygraphStream` consente di raccogliere queste voci di log delle modifiche per un grafo delle proprietà.

La funzionalità Neptune Streams deve essere abilitata sul cluster database Neptune. Per abilitare i flussi, imposta il parametro neptune_streams del cluster database su 1.

Consulta Acquisizione di modifiche al grafo in tempo reale mediante Neptune Streams.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:GetStreamRecords nel cluster.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta le seguenti azioni IAM, a seconda della query:

Nota che puoi limitare le query basate sul grafo delle proprietà utilizzando le seguenti chiavi contestuali IAM:

- neptune-db:QueryLanguage:Gremlin
- neptune-db:QueryLanguage:OpenCypher

Consulta Chiavi di condizioni disponibili nelle dichiarazioni delle policy di accesso ai dati IAM di Neptune.

Richiesta

- commitNum (nella CLI: `--commit-num`): un valore Long di tipo `long` (numero intero a 64 bit con segno).

Il numero di commit del record iniziale da leggere dal flusso di log delle modifiche. Questo parametro è obbligatorio se `iteratorType` è `AT_SEQUENCE_NUMBER` o `AFTER_SEQUENCE_NUMBER` e viene ignorato se `iteratorType` è `TRIM_HORIZON` o `LATEST`.

- encoding (nella CLI: `--encoding`): una codifica di tipo `string` (una stringa con codifica UTF-8).

  Se impostato su TRUE, Neptune comprime la risposta utilizzando la codifica gzip.

- iteratorType (nella CLI: `--iterator-type`): un valore IteratorType di tipo `string` (una stringa con codifica UTF-8).

  Può essere uno dei seguenti:

  - `AT_SEQUENCE_NUMBER`: indica che la lettura deve iniziare dal numero di sequenza di eventi specificato congiuntamente dai parametri `commitNum` e `opNum`.
  - `AFTER_SEQUENCE_NUMBER`: indica che la lettura deve iniziare immediatamente dopo il numero di sequenza di eventi specificato congiuntamente dai parametri `commitNum` e `opNum`.
  - `TRIM_HORIZON`: indica che la lettura deve iniziare dall'ultimo record non tagliato nel sistema, ovvero il record più vecchio non scaduto (non ancora eliminato) nel flusso di log delle modifiche.
  - `LATEST`: indica che la lettura deve iniziare dal record più recente non tagliato nel sistema, ovvero il record più recente non scaduto (non ancora eliminato) nel flusso di log delle modifiche.

- limit (nella CLI: `--limit`): un valore GetPropertygraphStreamInputLimitLong di tipo `long` (numero intero a 32 bit con segno), non meno di 1 o più di 100.000 ?st?s.

  Specifica il numero massimo di record da restituire. Esiste anche un limite di dimensioni di 10 MB per la risposta che non può essere modificato e che ha la precedenza sul numero di record specificato nel parametro `limit`. La risposta include un record che supera la soglia se il limite di 10 MB è stato raggiunto.

  L'intervallo per `limit` è compreso tra 1 e 100.000. Il valore predefinito è 10.

- opNum (nella CLI: `--op-num`): un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Il numero di sequenza dell'operazione all'interno del commit specificato da cui iniziare la lettura nei dati del flusso di log delle modifiche. Il valore predefinito è 1.

Risposta

- format: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

Formato di serializzazione per i record di modifica restituiti. Attualmente, l'unico valore supportato è `PG_JSON`.

- lastEventId: Obbligatorio: un array di mappa con coppie chiave-valore in cui:

  Ogni chiave è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Ogni valore è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Identificatore di sequenza dell'ultima modifica nella risposta del flusso.

  Un ID evento è composto da due campi: un `commitNum` che identifica una transazione che ha modificato il grafo e un `opNum` che identifica un'operazione specifica all'interno di tale transazione.

  Example

```
"eventId": {
          "commitNum": 12,
          "opNum": 1
        }
```

- lastTrxTimestampInMillis: Obbligatorio: un valore Long di tipo `long` (numero intero a 64 bit con segno).

  L'ora in cui è stato richiesto il commit per la transazione, in millisecondi dall'epoca Unix.

- records: Obbligatorio: un array di oggetti PropertygraphRecord.

  Un array di record serializzati del flusso di log delle modifiche inclusi nella risposta.

- totalRecords: Obbligatorio: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Il numero totale di record nella risposta.

Errori

- UnsupportedOperationException
- ExpiredStreamException
- InvalidParameterException
- MemoryLimitExceededException
- StreamRecordsNotFoundException

- ClientTimeoutException
- PreconditionsFailedException
- ThrottlingException
- ConstraintViolationException
- InvalidArgumentException
- IllegalArgumentException
- TooManyRequestsException

Strutture di dati dei flussi:

PropertygraphRecord (struttura)

Struttura di un record del grafo delle proprietà.

Campi

- commitTimestampInMillis: Obbligatorio: un valore Long di tipo `long` (numero intero a 64 bit con segno).

  L'ora in cui è stato richiesto il commit per la transazione, in millisecondi dall'epoca Unix.
- data: Obbligatorio: un oggetto PropertygraphData.

  Il record di modifica serializzato di Gremlin o openCypher.
- eventId: Obbligatorio: un array di mappa con coppie chiave-valore in cui:

  Ogni chiave è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Ogni valore è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore di sequenza del record di modifica del flusso.
- isLastOp: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Presente solo se questa operazione è l'ultima della transazione. Se è presente, è impostato su true. È utile per garantire che venga consumata un'intera transazione.
- op: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'operazione che ha creato la modifica.

PropertygraphData (struttura)

Un record di modifica di Gremlin o openCypher.

Campi

- from: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se si tratta di un arco (tipo = `e`), è l'ID del vertice `from` del nodo di origine corrispondente.
- id: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID dell'elemento Gremlin o openCypher.
- key: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della proprietà. Per le etichette degli elementi, questo è `label`.
- to: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Se si tratta di un arco (tipo = `e`), è l'ID del vertice `to` o del nodo di destinazione corrispondente.
- type: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di questo elemento Gremlin o openCypher. Deve essere un valore tra:
  - **v1**: etichetta di vertice per Gremlin o etichetta di nodo per openCypher.
  - **vp**: proprietà di vertice per Gremlin o proprietà di nodo per openCypher.
  - **e**: arco ed etichetta di arco per Gremlin o relazione e tipo di relazione per openCypher.
  - **ep**: proprietà di arco per Gremlin o proprietà di relazione per openCypher.
- value: Obbligatorio: un documento di tipo `document` (un contenuto aperto indipendente dal protocollo rappresentato da un modello di dati simile a JSON).

  Si tratta di un oggetto JSON che contiene un campo value per il valore stesso e un campo datatype per il tipo di dati JSON del valore:

  Example

```
"value": {
          "value": "(the new value"),
          "dataType": "(the JSON datatypenew value")
        }
```

API delle statistiche del piano dati e di riepilogo del grafo di Neptune

Azioni delle statistiche del grafo delle proprietà:

- GetPropertygraphStatistics (azione)
- ManagePropertygraphStatistics (azione)
- DeletePropertygraphStatistics (azione)
- GetPropertygraphSummary (azione)

Strutture delle statistiche:

- Statistics (struttura)
- StatisticsSummary (struttura)
- DeleteStatisticsValueMap (struttura)
- RefreshStatisticsIdMap (struttura)
- NodeStructure (struttura)
- EdgeStructure (struttura)
- SubjectStructure (struttura)
- PropertygraphSummaryValueMap (struttura)
- PropertygraphSummary (struttura)

GetPropertygraphStatistics (azione)

> Il nome AWS CLI per questa API è: `get-propertygraph-statistics`.

Ottiene le statistiche del grafo delle proprietà (Gremlin e openCypher).

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:GetStatisticsStatus nel cluster.

Richiesta

- Nessun parametro della richiesta.

Risposta

- payload: obbligatorio: un oggetto Statistiche.

  Statistiche per i dati del grafo delle proprietà.

- status: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice HTTP restituito della richiesta. Se la richiesta è riuscita, il codice è 200. Per un elenco degli errori più comuni, consulta Codici di errore comuni per la richiesta di statistiche DFE.

Errori

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ManagePropertygraphStatistics (azione)

Il nome AWS CLI per questa API è: `manage-propertygraph-statistics`.

Gestisce la generazione e l'uso delle statistiche del grafo delle proprietà.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:ManageStatistics nel cluster.

Richiesta

- mode (nella CLI: `--mode`): un valore StatisticsAutoGenerationMode di tipo `string` (una stringa con codifica UTF-8).

  La modalità di generazione delle statistiche. Una di `DISABLE_AUTOCOMPUTE`, `ENABLE_AUTOCOMPUTE` o`REFRESH`, l'ultimo dei quali attiva manualmente la generazione delle statistiche DFE.

Risposta

- payload: un oggetto RefreshStatisticsIdMap.

  Viene restituito solo per la modalità di aggiornamento.
- status: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice HTTP restituito della richiesta. Se la richiesta è riuscita, il codice è 200.

Errori

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

DeletePropertygraphStatistics (azione)

Il nome AWS CLI per questa API è: `delete-propertygraph-statistics`.

Elimina le statistiche per i dati di Gremlin e openCypher (grafo delle proprietà).

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:DeleteStatistics nel cluster.

Richiesta

- Nessun parametro della richiesta.

Risposta

- payload: un oggetto DeleteStatisticsValueMap.

  Payload dell'eliminazione.
- status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato di annullamento.
- statusCode: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Il codice di risposta HTTP: 200 se l'eliminazione è riuscita o 204 se non esistevano statistiche da eliminare.

Errori

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException

- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

GetPropertygraphSummary (azione)

Il nome AWS CLI per questa API è: `get-propertygraph-summary`.

Ottiene un riepilogo per un grafo delle proprietà.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:GetGraphSummary nel cluster.

Richiesta

- mode (nella CLI: `--mode`): un valore GraphSummaryType di tipo `string` (una stringa con codifica UTF-8).

  La modalità può avere uno dei due valori: `BASIC` (predefinito) e `DETAILED`.

Risposta

- payload: un oggetto PropertygraphSummaryValueMap.

  Payload contenente la risposta riassuntiva del grafo delle proprietà.
- statusCode: un numero intero di tipo `integer` (numero intero a 32 bit con segno).

  Il codice HTTP restituito della richiesta. Se la richiesta è riuscita, il codice è 200.

Errori

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException

- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Strutture delle statistiche:

Statistics (struttura)

Contiene informazioni sulle statistiche. Il motore DFE utilizza le informazioni sui dati del grafo Neptune per trovare compromessi efficaci al momento di pianificare l'esecuzione delle query. Queste informazioni assumono la forma di statistiche che includono i cosiddetti insiemi di caratteristiche, nonché statistiche sui predicati utili per la pianificazione delle query. Consulta Gestione delle statistiche utilizzabili dal motore DFE di Neptune.

Campi

- active: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se la generazione di statistiche DFE è abilitata o meno.
- autoCompute: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Indica se la generazione automatica di statistiche è abilitata o meno.
- date: questo è un valore SyntheticTimestamp_date_time di tipo `string` (una stringa con codifica UTF-8).

  L'ora UTC in cui sono state generate le statistiche DFE più recentemente.
- note: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

Una nota sui problemi nel caso in cui le statistiche non siano valide.

- signatureInfo: questo è un oggetto StatisticsSummary.

  Una struttura StatisticsSummary che contiene:

  - `signatureCount`: numero totale di firme in tutti i set di caratteristiche.
  - `instanceCount`: numero totale di istanze di set di caratteristiche.
  - `predicateCount`: numero totale di predicati univoci.
- statisticsId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Riporta l'ID dell'esecuzione corrente della generazione delle statistiche. Il valore -1 indica che non sono state generate statistiche.

StatisticsSummary (struttura)

Informazioni sui set di caratteristiche generati nelle statistiche.

Campi

- instanceCount: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

  : numero totale di istanze di set di caratteristiche.
- predicateCount: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

  Il numero totale di predicati univoci.
- signatureCount: questo è un numero intero di tipo `integer`(numero intero a 32 bit con segno).

  Il numero totale di firme in tutti i set di caratteristiche.

DeleteStatisticsValueMap (struttura)

Il payload per DeleteStatistics.

Campi

- active: questo è un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Lo stato corrente delle statistiche.

- statisticsId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID dell'esecuzione attualmente in corso per la generazione delle statistiche.

RefreshStatisticsIdMap (struttura)

Statistiche per la modalità `REFRESH`.

Campi

- statisticsId: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID dell'esecuzione attualmente in corso per la generazione delle statistiche.

NodeStructure (struttura)

Una struttura di nodi.

Campi

- count: questo è un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Numero di nodi con questa struttura specifica.
- distinctOutgoingEdgeLabels: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di etichette di archi in uscita distinte in questa struttura specifica.
- nodeProperties: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco delle proprietà dei nodi in questa struttura specifica.

EdgeStructure (struttura)

Una struttura di archi.

Campi

- count: questo è un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Numero di archi con questa struttura specifica.
- edgeProperties: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

Un elenco di proprietà di arco in questa struttura specifica.

SubjectStructure (struttura)

Una struttura di argomenti.

Campi

- count: questo è un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Numero di occorrenze di questa struttura specifica.
- predicates: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di predicati presenti in questa struttura specifica.

PropertygraphSummaryValueMap (struttura)

Payload per la risposta riassuntiva del grafo delle proprietà.

Campi

- graphSummary: questo è un oggetto PropertygraphSummary.

  Il riepilogo del grafo.
- lastStatisticsComputationTime: questo è un valore SyntheticTimestamp_date_time di tipo `string` (una stringa con codifica UTF-8).

  Il timestamp, in formato ISO 8601, dell'ora in cui Neptune ha calcolato le statistiche per l'ultima volta.
- version: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  La versione della risposta riassuntiva del grafo.

PropertygraphSummary (struttura)

L'API di riepilogo del grafo restituisce un elenco di sola lettura di etichette di nodi e archi, nonché di chiavi di proprietà, insieme al conteggio di nodi, archi e proprietà. Consulta Risposta riepilogativa per un grafo delle proprietà.

Campi

- edgeLabels: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di etichette di arco distinte nel grafo.

- edgeProperties: si tratta di oggetti LongValuedMap ed è un array di mappa con coppie chiave-valore in cui:

  Ogni chiave è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Ogni valore è un oggetto Long di tipo `long` (numero intero a 64 bit con segno).

  Un elenco di proprietà di arco distinte nel grafo, insieme al numero di archi in cui viene utilizzata ciascuna proprietà.

- edgeStructures: questo è un array di oggetti EdgeStructure.

  Questo campo è presente solo se la modalità richiesta è`DETAILED`. Contiene un elenco di strutture di archi.

- nodeLabels: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un elenco di etichette di nodi distinte nel grafo.

- nodeProperties: si tratta di oggetti LongValuedMap ed è un array di mappa con coppie chiave-valore in cui:

  Ogni chiave è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Ogni valore è un oggetto Long di tipo `long` (numero intero a 64 bit con segno).

  Il numero di proprietà di nodi distinte nel grafo.

- nodeStructures: questo è un array di oggetti NodeStructure.

  Questo campo è presente solo se la modalità richiesta è`DETAILED`. Contiene un elenco di strutture di nodi.

- numEdgeLabels: questo è un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Il numero di etichette di arco distinte nel grafo.

- numEdgeProperties: questo è un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Il numero di proprietà di arco distinte nel grafo.

- numEdges: questo è un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Il numero di archi nel grafo.
- numNodeLabels: questo è un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Il numero di etichette di nodi distinte nel grafo.
- numNodeProperties: questo è un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Un elenco di proprietà di nodi distinte nel grafo, insieme al numero di nodi in cui viene utilizzata ciascuna proprietà.
- numNodes: questo è un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Il numero di nodi nel grafo.
- totalEdgePropertyValues: questo è un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Il numero totale di utilizzi di tutte le proprietà di arco.
- totalNodePropertyValues: questo è un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Il numero totale di utilizzi di tutte le proprietà di nodi.

API di elaborazione dati Neptune ML

Azioni di elaborazione dati:

- StartMLDataProcessingJob (azione)
- ListMLDataProcessingJobs (azione)
- GetMLDataProcessingJob (azione)
- CancelMLDataProcessingJob (azione)

Strutture ML generiche:

- MlResourceDefinition (struttura)
- MlConfigDefinition (struttura)

StartMLDataProcessingJob (azione)

Il nome AWS CLI per questa API è: `start-ml-data-processing-job`.

Crea un nuovo processo di elaborazione dati Neptune ML per elaborare i dati dei grafi esportati da Neptune per l'addestramento. Consulta Comando `dataprocessing`.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:StartMLModelDataProcessingJob nel cluster.

Richiesta

- configFileName (nella CLI: `--config-file-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un file di specifiche dei dati che descrive come caricare i dati dei grafi esportati per l'addestramento. Il file viene generato automaticamente dal kit di strumenti di esportazione Neptune. Il valore predefinito è `training-data-configuration.json`.
- id (nella CLI: `--id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un identificatore univoco per il nuovo job. L'impostazione predefinita è un UUID generato automaticamente.
- inputDataS3Location (nella CLI: `--input-data-s3-location`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'URI della posizione Amazon S3 in cui SageMaker deve scaricare i dati necessari per eseguire il processo di elaborazione dati.
- modelType (nella CLI: `--model-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Uno dei due tipi di modello attualmente supportati da Neptune ML: modelli a grafo eterogenei (`heterogeneous`) e grafo della conoscenza (`kge`). Il valore predefinito è none. Se non specificato, Neptune ML sceglie automaticamente il tipo di modello in base ai dati.
- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

Il nome della risorsa Amazon (ARN) di un ruolo IAM che SageMaker può assumere per eseguire attività per tuo conto. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

- previousDataProcessingJobId (nella CLI: `--previous-data-processing-job-id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID del processo di elaborazione dati completato che è stato eseguito in una versione precedente dei dati.

- processedDataS3Location (nella CLI: `--processed-data-s3-location`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'URI della posizione Amazon S3 in cui SageMaker deve scaricare i risultati di un processo di elaborazione dati.

- processingInstanceType (nella CLI: `--processing-instance-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di istanza ML utilizzata durante l'elaborazione dati. La sua memoria deve essere sufficientemente grande da contenere il set di dati elaborato. L'impostazione predefinita è il tipo ml.r5 più piccolo, la cui memoria è dieci volte più grande della dimensione dei dati dei grafi esportati su disco.

- processingInstanceVolumeSizeInGB (nella CLI: `--processing-instance-volume-size-in-gb`): un valore Integer di tipo `integer` (numero intero a 32 bit con segno).

  La dimensione del volume del disco dell'istanza di elaborazione. Sia i dati di input che i dati elaborati vengono archiviati su disco, quindi le dimensioni del volume devono essere sufficientemente grandi da contenere entrambi i set di dati. Il valore predefinito è 0. Se non specificato o se pari a 0, Neptune ML sceglie automaticamente la dimensione del volume in base a quella dei dati.

- processingTimeOutInSeconds (nella CLI: `--processing-time-out-in-seconds`): un valore Integer di tipo `integer` (numero intero a 32 bit con segno).

  Timeout in secondi per il processo di elaborazione dati. Il valore predefinito è 86.400 (1 giorno).

- s3OutputEncryptionKMSKey (nella CLI: `--s-3-output-encryption-kms-key`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  La chiave Amazon Key Management Service (Amazon KMS) utilizzata da SageMaker per crittografare l'output del processo di elaborazione. Il valore predefinito è none.

- sagemakerIamRoleArn (nella CLI: `--sagemaker-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM per l'esecuzione di SageMaker. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.
- securityGroupIds (nella CLI: `--security-group-ids`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Gli ID del gruppo di sicurezza VPC. Il valore predefinito è None (Nessuno).
- subnets (nella CLI: `--subnets`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Gli ID delle sottoreti nel VPC Neptune. Il valore predefinito è None (Nessuno).
- volumeEncryptionKMSKey (nella CLI: `--volume-encryption-kms-key`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  La chiave Amazon Key Management Service (Amazon KMS) utilizzata da SageMaker per crittografare i dati nel volume di storage collegato alle istanze di calcolo ML che eseguono il job di addestramento. Il valore predefinito è None (Nessuno).

Risposta

- arn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN del processo di elaborazione dati.
- creationTimeInMillis: un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Il tempo impiegato per creare il nuovo processo di elaborazione, in millisecondi.
- id: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID univoco del nuovo processo di elaborazione dati.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException

- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

ListMLDataProcessingJobs (azione)

Il nome AWS CLI per questa API è: `list-ml-data-processing-jobs`.

Restituisce un elenco di processi di elaborazione dati Neptune ML. Consulta Elenco dei processi di elaborazione dati attivi utilizzando il comando dataprocessing di Neptune ML.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:ListMLDataProcessingJobs nel cluster.

Richiesta

- maxItems (nella CLI: `--max-items`): un valore ListMLDataProcessingJobsInputMaxItemsInteger di tipo `integer` (numero intero a 32 bit con segno), non meno di 1 o più di 1.024 ?st?s.

  Il numero massimo di elementi da restituire (da 1 a 1024; l'impostazione predefinita è 10).
- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Risposta

- ids: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Una pagina che elenca gli ID dei processi di elaborazione dati.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

GetMLDataProcessingJob (azione)

Il nome AWS CLI per questa API è: `get-ml-data-processing-job`.

Recupera le informazioni su un processo di elaborazione dati specificato. Consulta Comando `dataprocessing`.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:GetMLDataProcessingJobStatus nel cluster.

Richiesta

- id (nella CLI: `--id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco del processo di elaborazione dati da recuperare.
- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Risposta

- id: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco di questo processo di elaborazione dati.
- processingJob: un oggetto MlResourceDefinition.

  Definizione del processo di elaborazione dati.
- status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Stato del processo di elaborazione dati.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

CancelMLDataProcessingJob (azione)

Il nome AWS CLI per questa API è: `cancel-ml-data-processing-job`.

Annulla un processo di elaborazione dati Neptune ML. Consulta Comando `dataprocessing`.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:CancelMLDataProcessingJob nel cluster.

Richiesta

- clean (nella CLI: `--clean`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `TRUE`, questo flag specifica che tutti gli artefatti Neptune ML S3 devono essere eliminati quando il processo viene interrotto. Il valore predefinito è `FALSE`.
- id (nella CLI: `--id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco del processo di elaborazione dati.
- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Risposta

- status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato della richiesta di annullamento.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Strutture ML generiche:

MlResourceDefinition (struttura)

Definisce una risorsa Neptune ML.

Campi

- arn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN della risorsa.
- cloudwatchLogUrl: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'URL del log di CloudWatch per la risorsa.
- failureReason: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il motivo dell'eventuale errore.
- name: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della risorsa.
- outputLocation: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  La posizione di output.
- status: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato della risorsa.

MlConfigDefinition (struttura)

Contiene una configurazione Neptune ML.

Campi

- arn: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN della configurazione.
- name: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome della configurazione.

API di addestramento del modello Neptune ML

Azioni di addestramento del modello:

- StartMLModelTrainingJob (azione)
- ListMLModelTrainingJobs (azione)
- GetMLModelTrainingJob (azione)
- CancelMLModelTrainingJob (azione)

Strutture di addestramento del modello:

- CustomModelTrainingParameters (struttura)

StartMLModelTrainingJob (azione)

Il nome AWS CLI per questa API è: `start-ml-model-training-job`.

Crea un nuovo processo di addestramento del modello Neptune ML. Vedi Addestramento del modello mediante il comando `modeltraining`.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:StartMLModelTrainingJob nel cluster.

Richiesta

- baseProcessingInstanceType (nella CLI: `--base-processing-instance-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di istanza ML utilizzata nella preparazione e gestione dell'addestramento del modello ML. Si tratta di un'istanza CPU scelta in base ai requisiti di memoria per l'elaborazione dei dati e del modello di addestramento.
- customModelTrainingParameters (nella CLI: `--custom-model-training-parameters`): un oggetto CustomModelTrainingParameters.

  La configurazione per l'addestramento di modelli personalizzati. Questo è un oggetto JSON.
- dataProcessingJobId (nella CLI: `--data-processing-job-id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

L'ID del processo di elaborazione dati completato che ha creato i dati utilizzato durante l'addestramento.

- enableManagedSpotTraining (nella CLI: `--enable-managed-spot-training`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Ottimizza il costo per l'addestramento del modello di machine learning utilizzando le istanze spot di Amazon Elastic Compute Cloud. Il valore predefinito è `False`.

- id (nella CLI: `--id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un identificatore univoco per il nuovo job. L'impostazione predefinita è Un UUID generato automaticamente.

- maxHPONumberOfTrainingJobs (nella CLI: `--max-hpo-number-of-training-jobs`): un valore Integer di tipo `integer` (numero intero a 32 bit con segno).

  Numero totale massimo di processi di addestramento da avviare per il processo di ottimizzazione degli iperparametri. Il valore predefinito è 2. Neptune ML ottimizza automaticamente gli iperparametri del modello di machine learning. Per ottenere un modello che funzioni correttamente, utilizza almeno 10 processi (in altre parole, imposta `maxHPONumberOfTrainingJobs` su 10). In generale, maggiore è il numero di esecuzioni di ottimizzazione, migliori sono i risultati.

- maxHPOParallelTrainingJobs (nella CLI: `--max-hpo-parallel-training-jobs`): un valore Integer di tipo `integer` (numero intero a 32 bit con segno).

  Numero massimo di processi di addestramento paralleli da avviare per il processo di ottimizzazione degli iperparametri. Il valore predefinito è 2. Il numero di processi paralleli che è possibile eseguire è limitato dalle risorse disponibili sull'istanza di addestramento.

- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

- previousModelTrainingJobId (nella CLI: `--previous-model-training-job-id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID di un processo di addestramento del modello completato che si desidera aggiornare in modo incrementale in base ai dati aggiornati.

- s3OutputEncryptionKMSKey (nella CLI: `--s-3-output-encryption-kms-key`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  La chiave Amazon Key Management Service (KMS) utilizzata da SageMaker per crittografare l'output del job di elaborazione. Il valore predefinito è none.

- sagemakerIamRoleArn (nella CLI: `--sagemaker-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM per l'esecuzione di SageMaker. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

- securityGroupIds (nella CLI: `--security-group-ids`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Gli ID del gruppo di sicurezza VPC. Il valore predefinito è None (Nessuno).

- subnets (nella CLI: `--subnets`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Gli ID delle sottoreti nel VPC Neptune. Il valore predefinito è None (Nessuno).

- trainingInstanceType (nella CLI: `--training-instance-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di istanza ML utilizzato per l'addestramento del modello. Tutti i modelli Neptune ML supportano l'addestramento su CPU, GPU e multiGPU. Il valore predefinito è `ml.p3.2xlarge`. La scelta del tipo di istanza giusto per l'addestramento dipende dal tipo di attività, dalla dimensione del grafo e dal budget.

- trainingInstanceVolumeSizeInGB (nella CLI: `--training-instance-volume-size-in-gb`): un valore Integer di tipo `integer` (numero intero a 32 bit con segno).

  La dimensione del volume del disco dell'istanza di addestramento. Sia i dati di input che il modello di output vengono archiviati su disco, quindi le dimensioni del volume devono essere sufficientemente grandi da contenere entrambi i set di dati. Il valore predefinito è 0. Se non è specificato o è pari a 0, Neptune ML seleziona la dimensione del volume del disco in base al suggerimento generato nella fase di elaborazione dati.

- trainingTimeOutInSeconds (nella CLI: `--training-time-out-in-seconds`): un valore Integer di tipo `integer` (numero intero a 32 bit con segno).

  Timeout in secondi per il processo di addestramento. Il valore predefinito è 86.400 (1 giorno).

- trainModelS3Location (nella CLI: `--train-model-s3-location`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La posizione di Amazon S3 in cui devono essere archiviati gli artefatti del modello.

- volumeEncryptionKMSKey (nella CLI: `--volume-encryption-kms-key`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  La chiave Amazon Key Management Service (KMS) utilizzata da SageMaker per crittografare i dati nel volume di storage collegato alle istanze di calcolo ML che eseguono il job di addestramento. Il valore predefinito è None (Nessuno).

Risposta

- arn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN del processo di addestramento del nuovo modello.

- creationTimeInMillis: un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Il tempo per la creazione del processo di addestramento del modello, in millisecondi.

- id: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID univoco del nuovo processo di addestramento del modello.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException

- TooManyRequestsException

ListMLModelTrainingJobs (azione)

Il nome AWS CLI per questa API è: `list-ml-model-training-jobs`.

Elenca i processi di addestramento del modello Neptune ML. Vedi Addestramento del modello mediante il comando `modeltraining`.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:ListMLModelTrainingJobs nel cluster.

Richiesta

- maxItems (nella CLI: `--max-items`): un valore ListMLModelTrainingJobsInputMaxItemsInteger di tipo `integer` (numero intero a 32 bit con segno), non meno di 1 o più di 1.024 ?st?s.

  Il numero massimo di elementi da restituire (da 1 a 1024; l'impostazione predefinita è 10).
- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Risposta

- ids: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Una pagina con l'elenco degli ID dei processi di addestramento del modello.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException

- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

GetMLModelTrainingJob (azione)

Il nome AWS CLI per questa API è: `get-ml-model-training-job`.

Recupera informazioni su un processo di addestramento del modello Neptune ML. Vedi Addestramento del modello mediante il comando `modeltraining`.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:GetMLModelTrainingJobStatus in quel cluster.

Richiesta

- id (nella CLI: `--id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco del processo di addestramento del modello da recuperare.
- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Risposta

- hpoJob: un oggetto MlResourceDefinition.

  Il processo HPO.
- id: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco di questo processo di addestramento del modello.
- mlModels: una matrice di oggetti MlConfigDefinition.

Un elenco delle configurazioni dei modelli ML utilizzati.

- modelTransformJob: un oggetto MlResourceDefinition.

  Il processo di trasformazione del modello.

- processingJob: un oggetto MlResourceDefinition.

  Il processo di elaborazione dati.

- status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato del processo di addestramento del modello.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

CancelMLModelTrainingJob (azione)

Il nome AWS CLI per questa API è: `cancel-ml-model-training-job`.

Arresta un processo di addestramento del modello Neptune ML. Vedi Addestramento del modello mediante il comando `modeltraining`.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:CancelMLModelTrainingJob nel cluster.

Richiesta

- clean (nella CLI: `--clean`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `TRUE`, questo flag specifica che tutti gli artefatti Amazon S3 devono essere eliminati quando il processo viene interrotto. Il valore predefinito è `FALSE`.
- id (nella CLI: `--id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco del processo di addestramento del modello da annullare.
- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Risposta

- status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato dell'annullamento.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Strutture di addestramento del modello:

CustomModelTrainingParameters (struttura)

Contiene parametri personalizzati per l'addestramento del modello. Consulta Modelli personalizzati in Neptune ML.

Campi

- sourceS3DirectoryPath: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il percorso della posizione Amazon S3 in cui si trova il modulo Python che implementa il modello. Deve indicare una posizione Amazon S3 esistente valida che contenga almeno uno script di addestramento, uno script di trasformazione e un file `model-hpo-configuration.json`.
- trainingEntryPointScript: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del punto di ingresso nel modulo di uno script che esegue l'addestramento del modello e accetta gli iperparametri come argomenti della riga di comando, inclusi gli iperparametri fissi. Il valore predefinito è `training.py`.
- transformEntryPointScript: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del punto di ingresso nel modulo di uno script che deve essere eseguito dopo aver identificato il modello migliore ottenuto dalla ricerca degli iperparametri, in modo da calcolare gli artefatti del modello necessari per l'implementazione del modello. Deve poter essere eseguito senza argomenti della riga di comando. L'impostazione predefinita è `transform.py`.

API di trasformazione del modello Neptune ML

Azioni di trasformazione del modello:

- StartMLModelTransformJob (azione)
- ListMLModelTransformJobs (azione)
- GetMLModelTransformJob (azione)
- CancelMLModelTransformJob (azione)

Strutture di trasformazione del modello:

- CustomModelTransformParameters (struttura)

StartMLModelTransformJob (azione)

Il nome AWS CLI per questa API è: `start-ml-model-transform-job`.

Crea un nuovo processo di trasformazione del modello. Vedi Utilizzo di un modello addestrato per generare nuovi artefatti del modello.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:StartMLModelTransformJob nel cluster.

Richiesta

- baseProcessingInstanceType (nella CLI: `--base-processing-instance-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di istanza ML utilizzata nella preparazione e gestione dell'addestramento del modello ML. Si tratta di un'istanza di calcolo ML scelta in base ai requisiti di memoria per l'elaborazione dei dati e del modello di addestramento.
- baseProcessingInstanceVolumeSizeInGB (nella CLI: `--base-processing-instance-volume-size-in-gb`): un valore Integer di tipo `integer` (numero intero a 32 bit con segno).

  La dimensione del volume del disco dell'istanza di addestramento in gigabyte. Il valore predefinito è 0. Sia i dati di input che il modello di output vengono archiviati su disco, quindi le dimensioni del volume devono essere sufficientemente grandi da contenere entrambi i set di dati. Se non è specificato o è pari a 0, Neptune ML seleziona la dimensione del volume del disco in base al suggerimento generato nella fase di elaborazione dati.
- customModelTransformParameters (nella CLI: `--custom-model-transform-parameters`): un oggetto CustomModelTransformParameters.

  Informazioni di configurazione per una trasformazione del modello mediante un modello personalizzato. L'oggetto `customModelTransformParameters` contiene i seguenti campi, che devono avere valori compatibili con i parametri del modello salvati durante il processo di addestramento:
- dataProcessingJobId (nella CLI: `--data-processing-job-id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

L'ID di un processo di elaborazione dati completato. Devi includere `dataProcessingJobId` e un `mlModelTrainingJobId` o un `trainingJobName`.

- id (nella CLI: `--id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Un identificatore univoco per il nuovo job. L'impostazione predefinita è un UUID generato automaticamente.

- mlModelTrainingJobId (nella CLI: `--ml-model-training-job-id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID di un processo di addestramento del modello completato. Devi includere `dataProcessingJobId` e un `mlModelTrainingJobId` o un `trainingJobName`.

- modelTransformOutputS3Location (nella CLI: `--model-transform-output-s3-location`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  La posizione di Amazon S3 in cui devono essere archiviati gli artefatti del modello.

- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

- s3OutputEncryptionKMSKey (nella CLI: `--s-3-output-encryption-kms-key`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  La chiave Amazon Key Management Service (KMS) utilizzata da SageMaker per crittografare l'output del job di elaborazione. Il valore predefinito è none.

- sagemakerIamRoleArn (nella CLI: `--sagemaker-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM per l'esecuzione di SageMaker. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

- securityGroupIds (nella CLI: `--security-group-ids`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Gli ID del gruppo di sicurezza VPC. Il valore predefinito è None (Nessuno).

- subnets (nella CLI: `--subnets`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Gli ID delle sottoreti nel VPC Neptune. Il valore predefinito è None (Nessuno).

- trainingJobName (nella CLI: `--training-job-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome di un processo di addestramento SageMaker completato. Devi includere `dataProcessingJobId` e un `mlModelTrainingJobId` o un `trainingJobName`.
- volumeEncryptionKMSKey (nella CLI: `--volume-encryption-kms-key`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  La chiave Amazon Key Management Service (KMS) utilizzata da SageMaker per crittografare i dati nel volume di storage collegato alle istanze di calcolo ML che eseguono il job di addestramento. Il valore predefinito è None (Nessuno).

Risposta

- arn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN del processo di trasformazione del modello.
- creationTimeInMillis: un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Il tempo di creazione del processo di trasformazione del modello, in millisecondi.
- id: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID univoco del nuovo processo di trasformazione del modello.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException

- TooManyRequestsException

ListMLModelTransformJobs (azione)

Il nome AWS CLI per questa API è: `list-ml-model-transform-jobs`.

Restituisce un elenco di ID di processi di trasformazione del modello. Vedi Utilizzo di un modello addestrato per generare nuovi artefatti del modello.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:ListMLModelTransformJobs nel cluster.

Richiesta

- maxItems (nella CLI: `--max-items`): un valore ListMLModelTransformJobsInputMaxItemsInteger di tipo `integer` (numero intero a 32 bit con segno), non meno di 1 o più di 1.024 ?st?s.

  Il numero massimo di elementi da restituire (da 1 a 1024; l'impostazione predefinita è 10).
- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Risposta

- ids: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Una pagina dell'elenco degli ID di trasformazione del modello.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException

- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

GetMLModelTransformJob (azione)

Il nome AWS CLI per questa API è: `get-ml-model-transform-job`.

Ottiene informazioni su un processo di trasformazione del modello specificato. Vedi Utilizzo di un modello addestrato per generare nuovi artefatti del modello.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:GetMLModelTransformJobStatus in quel cluster.

Richiesta

- id (nella CLI: `--id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco del processo di trasformazione del modello da recuperare.
- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Risposta

- baseProcessingJob: un oggetto MlResourceDefinition.

  Il processo di elaborazione dati di base.
- id: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco del processo di trasformazione del modello da recuperare.

- models: una matrice di oggetti MlConfigDefinition.

  Un elenco delle informazioni di configurazione per i modelli utilizzati.

- remoteModelTransformJob: un oggetto MlResourceDefinition.

  Il processo di trasformazione del modello remoto.

- status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato del processo di trasformazione del modello.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

CancelMLModelTransformJob (azione)

Il nome AWS CLI per questa API è: `cancel-ml-model-transform-job`.

Annulla un processo di trasformazione del modello specificato. Vedi Utilizzo di un modello addestrato per generare nuovi artefatti del modello.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:CancelMLModelTransformJob nel cluster.

Richiesta

- clean (nella CLI: `--clean`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Se questo flag è impostato su `TRUE`, tutti gli artefatti Neptune ML S3 devono essere eliminati quando il job viene interrotto. Il valore predefinito è `FALSE`.
- id (nella CLI: `--id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID univoco del processo di trasformazione del modello da annullare.
- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Risposta

- status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato dell'annullamento.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Strutture di trasformazione del modello:

CustomModelTransformParameters (struttura)

Contiene parametri personalizzati per la trasformazione del modello. Vedi Utilizzo di un modello addestrato per generare nuovi artefatti del modello.

Campi

- sourceS3DirectoryPath: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il percorso della posizione Amazon S3 in cui si trova il modulo Python che implementa il modello. Deve indicare una posizione Amazon S3 esistente valida che contenga almeno uno script di addestramento, uno script di trasformazione e un file `model-hpo-configuration.json`.
- transformEntryPointScript: questa è una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il nome del punto di ingresso nel modulo di uno script che deve essere eseguito dopo aver identificato il modello migliore ottenuto dalla ricerca degli iperparametri, in modo da calcolare gli artefatti del modello necessari per l'implementazione del modello. Deve poter essere eseguito senza argomenti della riga di comando. Il valore predefinito è `transform.py`.

API dell'endpoint di inferenza Neptune ML

Azioni dell'endpoint di inferenza:

- CreateMLEndpoint (azione)
- ListMLEndpoints (azione)
- GetMLEndpoint (azione)
- DeleteMLEndpoint (azione)

CreateMLEndpoint (azione)

Il nome AWS CLI per questa API è: `create-ml-endpoint`.

Crea un nuovo endpoint di inferenza Neptune ML che consente di eseguire query su un modello specifico creato dal processo di addestramento del modello. Consulta Gestione degli endpoint di inferenza mediante il comando endpoints.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:CreateMLEndpoint nel cluster.

Richiesta

- id (nella CLI: `--id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco per l'endpoint. L'impostazione predefinita è un nome con timestamp generato automaticamente.
- instanceCount (nella CLI: `--instance-count`): un valore Integer di tipo `integer` (numero intero a 32 bit con segno).

  Il numero minimo di istanze Amazon EC2 da implementare su un endpoint ai fini della previsione. Il valore predefinito è 1.
- instanceType (nella CLI: `--instance-type`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il tipo di istanza Neptune ML da utilizzare per la manutenzione online. Il valore predefinito è `ml.m5.xlarge`. La scelta dell'istanza ML per un endpoint di inferenza dipende dal tipo di attività, dalla dimensione del grafo e dal budget.
- mlModelTrainingJobId (nella CLI: `--ml-model-training-job-id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID del processo di addestramento del modello completato che ha creato il modello a cui punterà l'endpoint di inferenza. Devi fornire `mlModelTrainingJobId` o `mlModelTransformJobId`.
- mlModelTransformJobId (nella CLI: `--ml-model-transform-job-id`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID del processo di trasformazione del modello completato. Devi fornire `mlModelTrainingJobId` o `mlModelTransformJobId`.
- modelName (nella CLI: `--model-name`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  Tipo di modello per l'addestramento. Per impostazione predefinita, il modello Neptune ML si basa automaticamente sul `modelType` utilizzato nell'elaborazione dati, ma è possibile specificare un tipo di modello diverso qui. L'impostazione predefinita è `rgcn` per i grafi eterogenei e `kge` per i grafi

della conoscenza. L'unico valore valido per grafi eterogenei è `rgcn`. I valori validi per i grafi della conoscenza sono `kge`, `transe`, `distmult` e `rotate`.

- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.
- update (nella CLI: `--update`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Se impostato su `true`, `update` indica che si tratta di una richiesta di aggiornamento. Il valore predefinito è `false`. Devi fornire `mlModelTrainingJobId` o `mlModelTransformJobId`.
- volumeEncryptionKMSKey (nella CLI: `--volume-encryption-kms-key`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  La chiave Amazon Key Management Service (Amazon KMS) utilizzata da SageMaker per crittografare i dati nel volume di storage collegato alle istanze di calcolo ML che eseguono il job di addestramento. Il valore predefinito è None (Nessuno).

Risposta

- arn: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN per il nuovo endpoint di inferenza.
- creationTimeInMillis: un valore Long di tipo `long` (numero intero a 64 bit con segno).

  Il tempo di creazione dell'endpoint, in millisecondi.
- id: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID univoco dell'endpoint.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException

- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

ListMLEndpoints (azione)

Il nome AWS CLI per questa API è: `list-ml-endpoints`.

Elenca gli endpoint di inferenza esistenti. Consulta Gestione degli endpoint di inferenza mediante il comando endpoints.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:ListMLEndpoints nel cluster.

Richiesta

- maxItems (nella CLI: `--max-items`): un valore ListMLEndpointsInputMaxItemsInteger di tipo `integer` (numero intero a 32 bit con segno), non meno di 1 o più di 1.024 ?st?s.

  Il numero massimo di elementi da restituire (da 1 a 1024; l'impostazione predefinita è 10).
- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Risposta

- ids: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Una pagina dell'elenco di ID degli endpoint di inferenza.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

GetMLEndpoint (azione)

Il nome AWS CLI per questa API è: `get-ml-endpoint`.

Recupera i dettagli su un endpoint di inferenza. Consulta Gestione degli endpoint di inferenza mediante il comando endpoints.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:GetMLEndpointStatus in quel cluster.

Richiesta

- id (nella CLI: `--id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco per l'endpoint.
- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Risposta

- endpoint: un oggetto MlResourceDefinition.

  La definizione dell'endpoint.
- endpointConfig: un oggetto MlConfigDefinition.

  La configurazione dell'endpoint.
- id: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco per l'endpoint.
- status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato dell'endpoint di inferenza.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

DeleteMLEndpoint (azione)

Il nome AWS CLI per questa API è: `delete-ml-endpoint`.

Annulla la creazione di un endpoint di inferenza Neptune ML. Consulta Gestione degli endpoint di inferenza mediante il comando endpoints.

Quando si richiama questa operazione in un cluster Neptune in cui è abilitata l'autenticazione IAM, all'utente o al ruolo IAM che effettua la richiesta deve essere associata una policy che consenta l'azione IAM neptune-db:DeleteMLEndpoint nel cluster.

Richiesta

- clean (nella CLI: `--clean`): un valore booleano di tipo `boolean` [un valore booleano (vero o falso)].

  Se questo flag è impostato su `TRUE`, tutti gli artefatti Neptune ML S3 devono essere eliminati quando il job viene interrotto. Il valore predefinito è `FALSE`.
- id (nella CLI: `--id`): Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'identificatore univoco per l'endpoint.
- neptuneIamRoleArn (nella CLI: `--neptune-iam-role-arn`): una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ARN di un ruolo IAM che fornisce a Neptune l'accesso alle risorse SageMaker e Amazon S3. Deve essere elencato nel gruppo di parametri del cluster database o si verificherà un errore.

Risposta

- status: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Lo stato dell'annullamento.

Errori

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

- IllegalArgumentException
- TooManyRequestsException

Eccezioni dell'API del piano dati Neptune

Eccezioni:

- AccessDeniedException (struttura)
- BadRequestException (struttura)
- BulkLoadIdNotFoundException (struttura)
- CancelledByUserException (struttura)
- ClientTimeoutException (struttura)
- ConcurrentModificationException (struttura)
- ConstraintViolationException (struttura)
- ExpiredStreamException (struttura)
- FailureByQueryException (struttura)
- IllegalArgumentException (struttura)
- InternalFailureException (struttura)
- InvalidArgumentException (struttura)
- InvalidNumericDataException (struttura)
- InvalidParameterException (struttura)
- LoadUrlAccessDeniedException (struttura)
- MalformedQueryException (struttura)
- MemoryLimitExceededException (struttura)
- MethodNotAllowedException (struttura)
- MissingParameterException (struttura)
- MLResourceNotFoundException (struttura)
- ParsingException (struttura)
- PreconditionsFailedException (struttura)
- QueryLimitExceededException (struttura)
- QueryLimitException (struttura)

- QueryTooLargeException (struttura)
- ReadOnlyViolationException (struttura)
- S3Exception (struttura)
- ServerShutdownException (struttura)
- StatisticsNotAvailableException (struttura)
- StreamRecordsNotFoundException (struttura)
- ThrottlingException (struttura)
- TimeLimitExceededException (struttura)
- TooManyRequestsException (struttura)
- UnsupportedOperationException (struttura)
- UnloadUrlAccessDeniedException (struttura)

AccessDeniedException (struttura)

Generato in caso di errore di autenticazione o autorizzazione.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

BadRequestException (struttura)

Generato quando viene inviata una richiesta che non può essere elaborata.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

Il codice di stato HTTP restituito con l'eccezione.

- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta on valida.

BulkLoadIdNotFoundException (struttura)

Generato quando non è possibile trovare l'ID di un processo di caricamento in blocco specificato.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID del processo di caricamento in blocco che non è stato possibile trovare.

CancelledByUserException (struttura)

Generato quando un utente ha annullato una richiesta.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

L'ID della richiesta in questione.

ClientTimeoutException (struttura)

Generato quando una richiesta è scaduta nel client.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

ConcurrentModificationException (struttura)

Generato quando una richiesta tenta di modificare dati che vengono modificati contemporaneamente da un altro processo.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

ConstraintViolationException (struttura)

Generato quando un valore in un campo di richiesta non soddisfa i vincoli richiesti.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

ExpiredStreamException (struttura)

Generato quando una richiesta tenta di accedere a un flusso scaduto.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

FailureByQueryException (struttura)

Generato quando una richiesta non riesce.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.

- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

IllegalArgumentException (struttura)

Generato quando un argomento in una richiesta non è supportato.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

InternalFailureException (struttura)

Generato quando l'elaborazione della richiesta non è riuscita in modo imprevisto.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

InvalidArgumentException (struttura)

Generato quando un argomento in una richiesta ha un valore non valido.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

InvalidNumericDataException (struttura)

Generato quando vengono rilevati dati numerici non validi durante la gestione di una richiesta.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

InvalidParameterException (struttura)

Generato quando il valore di un parametro non è valido.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.

- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta che include un parametro non valido.

LoadUrlAccessDeniedException (struttura)

Generato quando viene negato l'accesso a un URL di caricamento specificato.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

MalformedQueryException (struttura)

Generato quando viene inviata una query sintatticamente errata o che non supera una convalida aggiuntiva.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta di query non valida.

MemoryLimitExceededException (struttura)

Generato quando una richiesta non riesce perché le risorse di memoria sono insufficienti. È possibile provare di nuovo a eseguire la richiesta.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta non riuscita.

MethodNotAllowedException (struttura)

Generato quando il metodo HTTP utilizzato da una richiesta non è supportato dall'endpoint in uso.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

MissingParameterException (struttura)

Generato quando manca un parametro obbligatorio.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

Il codice di stato HTTP restituito con l'eccezione.

- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in cui manca il parametro.

MLResourceNotFoundException (struttura)

Generato quando non è stato possibile trovare una risorsa di machine learning specificata.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

ParsingException (struttura)

Generato quando si verifica un problema di analisi.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

L'ID della richiesta in questione.

PreconditionsFailedException (struttura)

Generato quando una condizione preliminare per l'elaborazione di una richiesta non è soddisfatta.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

QueryLimitExceededException (struttura)

Generato quando il numero di query attive supera quello che il server può elaborare. È possibile provare a eseguire di nuovo la query in questione quando il sistema è meno occupato.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta che ha superato il limite.

QueryLimitException (struttura)

Generato quando la dimensione di una query supera il limite di sistema.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta che ha superato il limite.

QueryTooLargeException (struttura)

Generato quando il testo di una query è troppo grande.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta troppo grande.

ReadOnlyViolationException (struttura)

Generato quando una richiesta tenta di scrivere su una risorsa di sola lettura.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.

- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in cui manca il parametro.

S3Exception (struttura)

Generato quando c'è un problema di accesso ad Amazon S3.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

ServerShutdownException (struttura)

Generato quando il server si arresta durante l'elaborazione di una richiesta.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

StatisticsNotAvailableException (struttura)

Generato quando non sono disponibili le statistiche necessarie per soddisfare una richiesta.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

StreamRecordsNotFoundException (struttura)

Generato quando non è possibile trovare i record di flusso richiesti da una query.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

ThrottlingException (struttura)

Generato se la frequenza delle richieste supera la velocità di trasmissione effettiva massima. È possibile provare a eseguire di nuovo le richieste dopo aver riscontrato questa eccezione.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

Il messaggio dettagliato che descrive il problema.

- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta che non è stato possibile elaborare per il motivo indicato.

TimeLimitExceededException (struttura)

Generato quando un'operazione supera il limite di tempo consentito.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta che non è stato possibile elaborare per il motivo indicato.

TooManyRequestsException (struttura)

Generato quando il numero di richieste in fase di elaborazione supera il limite.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta che non è stato possibile elaborare per il motivo indicato.

UnsupportedOperationException (struttura)

Generato quando una richiesta tenta di avviare un'operazione che non è supportata.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

UnloadUrlAccessDeniedException (struttura)

Generato quando viene negato l'accesso a un URL che è una destinazione di scaricamento.

Campi

- code: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il codice di stato HTTP restituito con l'eccezione.
- detailedMessage: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  Il messaggio dettagliato che descrive il problema.
- requestId: Obbligatorio: una stringa di tipo `string` (una stringa con codifica UTF-8).

  L'ID della richiesta in questione.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.