aws

Entwicklerhandbuch

Amazon Quantum Ledger Database (Amazon QLDB)

Amazon Quantum Ledger Database (Amazon QLDB): Entwicklerhandbuch

Table of Contents

Was ist Amazon QLDB?

Amazon Quantum Ledger Database (Amazon QLDB) ist eine vollständig verwaltete Ledger-Datenbank, die ein transparentes, unveränderliches und kryptografisch überprüfbares Transaktionsprotokoll bereitstellt, das einer zentralen vertrauenswürdigen Stelle gehört. Mit Amazon QLDB können Sie alle Änderungen der Anwendungsdaten verfolgen und einen vollständigen und überprüfbaren Verlauf der Änderungen im Laufe der Zeit pflegen. Weitere Informationen zu den verschiedenen Datenbankoptionen, die auf Amazon Web Services verfügbar sind, finden Sie unter Auswahl der richtigen Datenbank für Ihr Unternehmen aufAWS.

Ein Ledger (Hauptbuch) wird normalerweise verwendet, um einen Verlauf der wirtschaftlichen und finanziellen Aktivitäten eines Unternehmens aufzuzeichnen. Viele Organisationen erstellen Anwendungen mit Ledger-artigen Funktionen, da sie einen korrekten Verlauf der Daten ihrer Anwendungen beibehalten möchten. Beispiel: Sie möchten möglicherweise den Verlauf von Haben und Soll in Bankgeschäften verfolgen, die Herkunft der Daten in einem Versicherungsanspruch überprüfen oder die Bewegung eines Elements in einem Lieferkettennetzwerk verfolgen. Ledger-Anwendungen werden häufig mithilfe von Prüfungstabellen oder Prüfpfaden implementiert, die in relationalen Datenbanken erstellt werden.

Amazon QLDB ist eine neue Datenbankklasse, mit der Sie den komplexen Entwicklungsaufwand für die Erstellung eigener Ledger-ähnlicher Anwendungen vermeiden können. Mit QLDB ist der Verlauf der Änderungen an Ihren Daten unveränderlich — er kann nicht überschrieben oder an Ort und Stelle geändert werden. Und mithilfe von Kryptografie können Sie überprüfen, ob keine unbeabsichtigten Änderungen an den Daten Ihrer Anwendung vorgenommen wurden. QLDB verwendet ein unveränderliches Transaktionsprotokoll, ein sogenanntes Journal. Im Journal wird nur angehängt und es besteht aus einer sequenzierten und hashverketteten Gruppe von Blöcken, die Ihre festgeschriebenen Daten enthalten.

Amazon QLDB-Video

Einen Überblick über Amazon QLDB und darüber, wie Sie davon profitieren können, finden Sie in diesem QLDB-Übersichtsvideo unter YouTube.

Amazon QLDB-Preise

Mit Amazon QLDB zahlen Sie nur das, was Sie tatsächlich nutzen, ohne Mindestgebühren oder eine obligatorische Nutzung des Dienstes. Sie zahlen nur für die Ressourcen, die Ihre Ledger-Datenbank nutzt, und müssen nicht im Vorhinein bereitstellen.

Weitere Informationen finden Sie unter Amazon QLDB-Preise.

Erste Schritte mit QLDB

Wir empfehlen, dass Sie zuerst die folgenden Themen lesen:

- Übersicht über Amazon QLDB— Um einen hochgradigen Überblick über QLDB zu erhalten.
- Kernkonzepte und Terminologie in Amazon QLDB— Um grundlegende QLDB-Konzepte und -Terminologie zu lernen.
- Zugreifen auf Amazon QLDB— Um zu erfahren, wie Sie mit derAWS Management Console, API oderAWS Command Line Interface (AWS CLI) auf QLDB zugreifen können.
- Funktionsweise von Amazon QLDB mit IAM— Um zu erfahren, wie Sie den Zugriff auf QLDB mithilfe vonAWS Identity and Access Management (IAM) steuern.

Informationen dazu, wie Sie schnell mit der QLDB-Konsole beginnen können, finden Sie unterErste Schritte mit der Amazon QLDB-Konsole.

Weitere Informationen zur Entwicklung mit QLDB mithilfe einesAWS bereitgestellten Treibers finden Sie unterErste Schritte mit dem Amazon-QLDB-Treiber.

Übersicht über Amazon QLDB

Die folgenden Abschnitte vermitteln eine allgemeine Übersicht über Amazon-QLDB-Servicekomponenten und wie sie interagieren.

Themen

- Zuerst Tagebuch
- Immutable (Unveränderlich)
- Kryptografisch überprüfbar
- SQL-ähnlich und dokumentenflexibel

- Open-Source-Entwicklertools
- Serverlos und hochverfügbar
- Qualität für Unternehmen

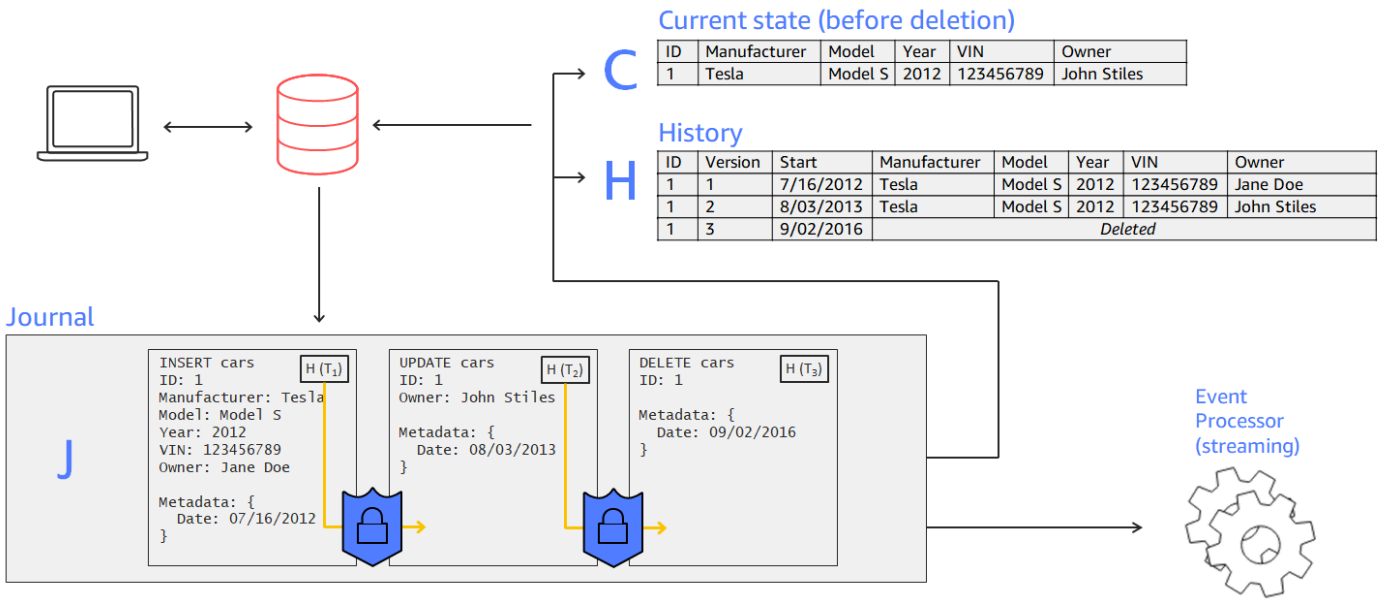
Zuerst Tagebuch

In der traditionellen Datenbankarchitektur schreiben Sie im Allgemeinen Daten in Tabellen als Teil einer Transaktion. Ein Transaktionsprotokoll — in der Regel eine interne Implementierung — zeichnet alle Transaktionen und die von ihnen vorgenommenen Datenbankänderungen auf. Das Transaktionsprotokoll ist eine zentrale Komponente der Datenbank. Sie benötigen das Protokoll, um Transaktionen im Falle eines Systemausfalls, einer Notfallwiederherstellung oder einer Datenreplikation wiederzugeben. Datenbanktransaktionsprotokolle sind jedoch nicht unveränderlich und nicht darauf ausgelegt, Benutzern direkten und einfachen Zugriff zu bieten.

In Amazon QLDB ist das Journal der Kern der Datenbank. Das Journal ähnelt strukturell einem Transaktionslog und ist eine unveränderliche, nur angehängte Datenstruktur, in der Ihre Anwendungsdaten zusammen mit den zugehörigen Metadaten gespeichert werden. Alle Schreibtransaktionen, einschließlich Aktualisierungen und Löschungen, werden zuerst im Journal festgeschrieben.

QLDB verwendet das Journal, um den aktuellen Status Ihrer Ledger-Daten zu ermitteln, indem es diese in abfragbaren, benutzerdefinierten Tabellen materialisiert. Diese Tabellen bieten außerdem einen zugänglichen Verlauf aller Transaktionsdaten, einschließlich Dokumentrevisionen und Metadaten. Darüber hinaus verarbeitet das Journal Gleichzeitigkeit, Sequenzierung, kryptografische Verifizierung und Verfügbarkeit der Ledger-Daten.

Das folgende Diagramm zeigt die QLDB-Journalarchitektur.



- In diesem Beispiel stellt eine Anwendung eine Verbindung zu einem Ledger her und führt Transaktionen aus, mit denen ein Dokument in eine Tabelle mit dem Namen eingefügt, aktualisiert und gelöscht wird`cars`.
- Die Daten werden nacheinander zuerst in das Journal geschrieben.
- Dann werden die Daten in die Tabelle mit integrierten Ansichten materialisiert. Mit diesen Ansichten können Sie sowohl den aktuellen Zustand als auch den vollständigen Verlauf des Fahrzeugs abfragen, wobei jeder Revision eine Versionsnummer zugewiesen ist.
- Sie können Daten auch direkt aus dem Journal exportieren oder streamen.

Immutable (Unveränderlich)

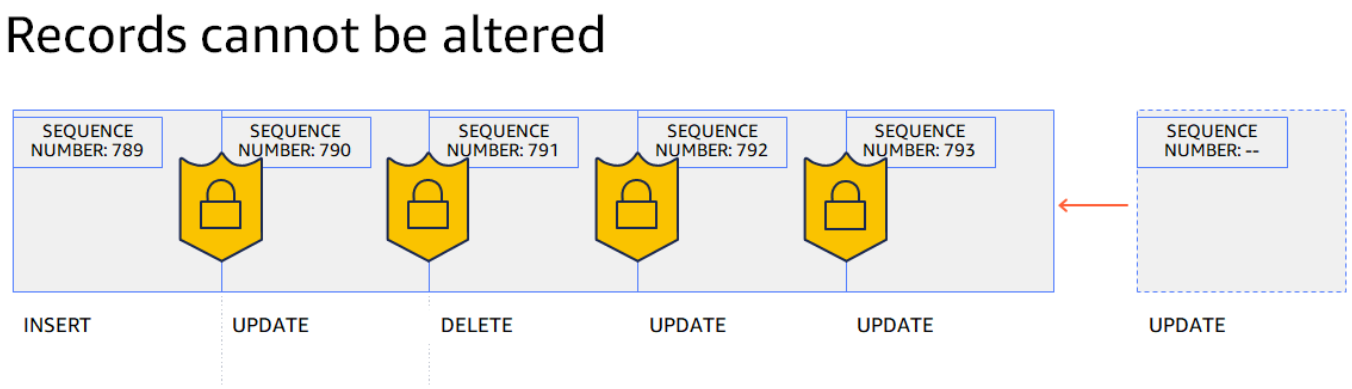
Da das QLDB-Journal nur angehängt werden kann, zeichnet es alle Änderungen an Ihren Daten, die nicht geändert oder überschrieben werden können, vollständig auf. Es gibt keine APIs oder andere Methoden, um übermittelte Daten zu ändern. Mit dieser Journalstruktur können Sie auf den vollständigen Verlauf Ihres Hauptbuchs zugreifen und ihn abfragen.

> **Note**
>
> Die einzige Ausnahme von der Unveränderlichkeit, die QLDB unterstützt, ist die Datenredaktion. Mit dieser Funktion können Sie gesetzliche Bestimmungen wie die

Allgemeine Datenschutzverordnung (DSGVO) in der Europäischen Union und den California Consumer Privacy Act (CCPA) einhalten.
QLDB bietet eine Schwärzungsoperation, mit der Sie inaktive Dokumentrevisionen in der Historie einer Tabelle dauerhaft löschen können. Dieser Vorgang löscht nur die Benutzerdaten in der angegebenen Revision und lässt die Journalsequenz und die Dokumentmetadaten unverändert. Dadurch wird die allgemeine Datenintegrität Ihres Ledgers gewahrt. Weitere Informationen finden Sie unter Redigieren von Dokumentrevisionen.

QLDB schreibt in einer Transaktion einen Block in das Journal. Jeder Block enthält Eintragsobjekte, die die Dokumente darstellen, die Sie einfügen, aktualisieren und löschen, sowie die Anweisungen, die Sie ausgeführt haben, um sie zu speichern. Diese Blöcke werden sequenziert und Hash-verkettet, um die Datenintegrität zu gewährleisten.

Das folgende Diagramm veranschaulicht diese Journalstruktur.

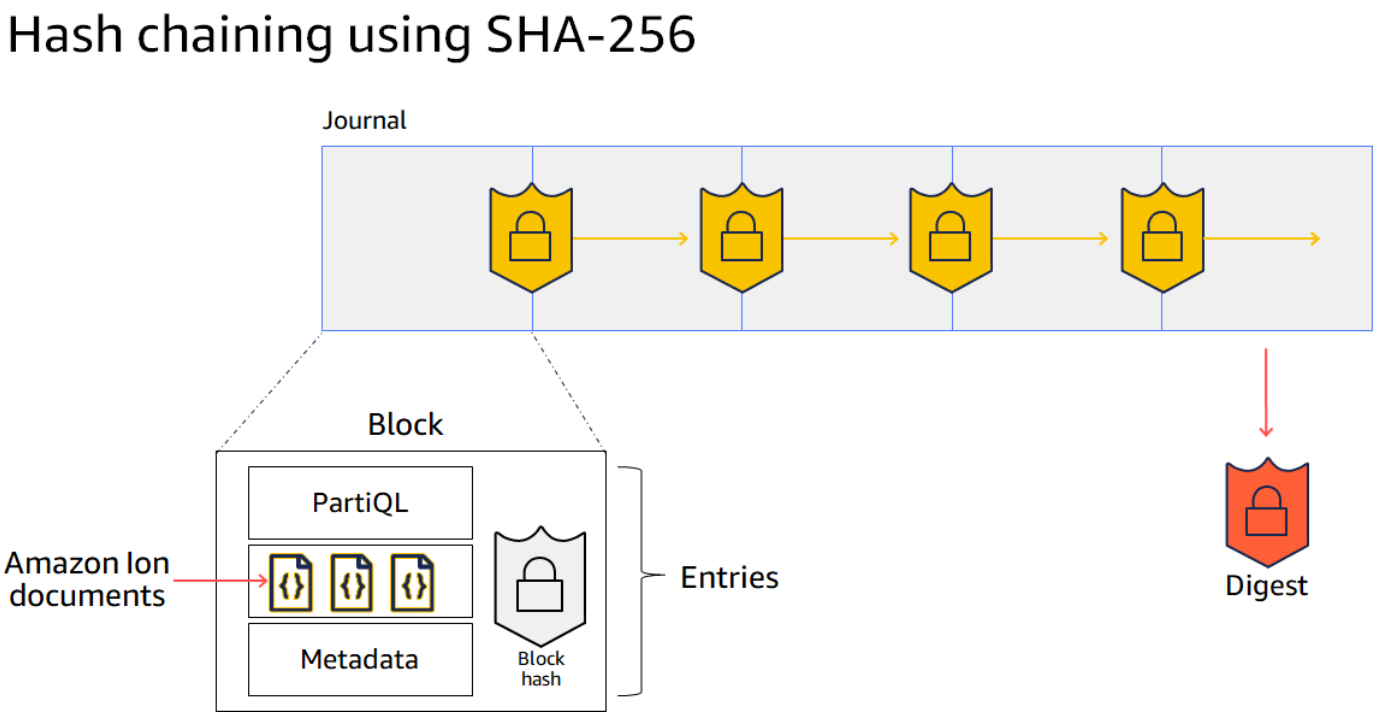


Das Diagramm zeigt, dass Transaktionen an das Journal als Blöcke übergeben werden, die zur Verifizierung Hash-verkettet sind. Jeder Block verfügt über eine Sequenznummer, um seine Adresse anzugeben.

Kryptografisch überprüfbar

Journalblöcke werden sequenziert und mit kryptografischen Hashing-Techniken verkettet, so ähnlich wie Blockchains. QLDB verwendet die Hash-Kette des Journals, um die Integrität der Transaktionsdaten mithilfe einer kryptografischen Überprüfungsmethode zu gewährleisten. Mit einem Digest (einem Hash-Wert, der die vollständige Hash-Kette eines Journals ab einem Zeitpunkt darstellt) und einem Hash-Prüfnachweis (einem Mechanismus, der die Gültigkeit eines beliebigen Knotens innerhalb eines binären Hash-Baums belegt) können Sie sicherstellen, dass an Ihren Daten keine unbeabsichtigten Änderungen vorgenommen wurden.

Das folgende Diagramm zeigt einen Digest, der die vollständige Hash-Kette eines Journals zu einem bestimmten Zeitpunkt abdeckt.



In diesem Diagramm werden die Journalblöcke mit der kryptografischen Hash-Funktion SHA-256 gehasht und nacheinander mit den folgenden Blöcken verkettet. Jeder Block enthält Einträge, die Ihre Datendokumente, Metadaten und die PartiQL-Anweisungen enthalten, die in der Transaktion ausgeführt wurden.

Weitere Informationen finden Sie unter Datenverifizierung in Amazon QLDB.

SQL-ähnlich und dokumentenflexibel

QLDB verwendet PartiQL als Abfragesprache und Amazon Ion als dokumentorientiertes Datenmodell. PartiQL ist eine quelloffene, SQL-kompatible Abfragesprache, die für die Arbeit mit Ion erweitert wurde. Mit PartiSQL können Sie Ihre Daten mit vertrauten SQL-Operatoren einfügen, abfragen und verwalten. Beim Abfragen von flachen Dokumenten entspricht die Syntax der Verwendung von SQL zum Abfragen relationaler Tabellen. Weitere Informationen zur QLDB-Implementierung von PartiQL finden Sie imAmazon QLDB PartiQL-Referenz.

Amazon Ion ist eine Obermenge von JSON. Ion ist ein dokumentenbasiertes Open-Source-Datenformat, das Ihnen die Flexibilität bietet, strukturierte, halbstrukturierte und verschachtelte Daten zu speichern und zu verarbeiten. Weitere Informationen zu Ion in QLDB finden Sie imReferenz zum Amazon Ion-Datenformat in Amazon QLDB.

Einen allgemeinen Vergleich der Kernkomponenten und Funktionen herkömmlicher relationaler Datenbanken mit QLDB finden Sie unterVom relationalen zum Ledger.

Open-Source-Entwicklertools

Um die Anwendungsentwicklung zu vereinfachen, bietet QLDB Open-Source-Treiber in verschiedenen Programmiersprachen. Sie können diese Treiber verwenden, um mit der Transaktionsdaten-API zu interagieren, indem Sie PartiQL-Anweisungen in einem Ledger ausführen und die Ergebnisse dieser Anweisungen verarbeiten. Informationen und Tutorials zu den aktuell unterstützten Treibersprachen finden Sie unter Erste Schritte mit dem Amazon-QLDB-Treiber.

Amazon Ion bietet auch Clientbibliotheken, die Ion-Daten für Sie verarbeiten. Entwicklerhandbücher und Codebeispiele für die Verarbeitung von Ion-Daten finden Sie in der Amazon Ion-Dokumentation unter GitHub.

Serverlos und hochverfügbar

QLDB ist vollständig verwaltet, serverlos und hochverfügbar. Der Service wird automatisch skaliert, um die Anforderungen Ihrer Anwendung zu erfüllen, und Sie müssen keine Instanzen oder Kapazität bereitstellen. Mehrere Kopien Ihrer Daten werden innerhalb einer Availability Zone und zwischen Availability Zones in einer repliziertAWS-Region.

Qualität für Unternehmen

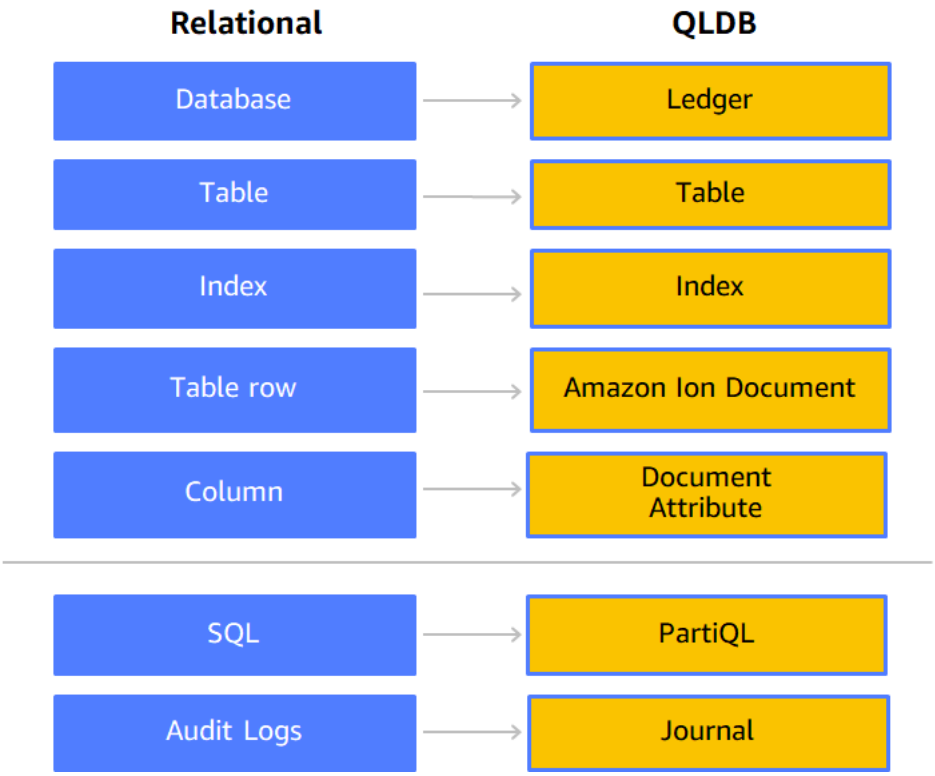
QLDB-Transformationale entsprechen in vollem Umfang die ACID (ACID), Consistency [Konsistenz], Consistency, Consistency, Consistency, Consistency, Consistency QLDB verwendet Optimist Concurrency Control (OCC), und Transaktionen werden mit voller Serialisierbarkeit ausgeführt — der höchsten Isolationsstufe. Dies bedeutet, dass keine Gefahr besteht, dass Phantomlesevorgänge, fehlerhafte Lesevorgänge, Schreibverzerrungen oder andere ähnliche Probleme mit der Gleichzeitigkeit auftreten. Weitere Informationen finden Sie unter Amazon QLDB-Nebenläufigkeit von Amazon QLDB-Nebenläufigkeit.

Vom relationalen zum Ledger

Wenn Sie Anwendungsentwickler sind, haben Sie möglicherweise einige Erfahrung mit einem relationalen Datenbankmanagementsystem (RDBMS) und der Structured Query Language (SQL). Wenn Sie mit Amazon QLDB arbeiten, werden Sie auf viele Gemeinsamkeiten stoßen. Wenn Sie sich weiterführenden Themen zuwenden, werden Sie auch auf leistungsstarke neue Funktionen stoßen, die QLDB auf der RDBMS-Grundlage aufgebaut hat. In diesem Abschnitt werden gängige

Datenbankkomponenten und -Operationen sowie Vergleiche und Gegenüberstellungen mit ihren Äquivalenten in QLDB verglichen.

Das folgende Diagramm zeigt die Mapping-Konstrukte der Kernkomponenten zwischen einem herkömmlichen RDBMS und Amazon QLDB.



Die folgende Tabelle zeigt die wichtigsten Gemeinsamkeiten und Unterschiede der integrierten Betriebsfunktionen zwischen einem herkömmlichen RDBMS und QLDB.

Operation	RDBMS	QLDB
Erstellen von Tabellen	`CREATE TABLE`-Anweisung, die alle Spaltennamen und Datentypen definiert	`CREATE TABLE`-Anweisung, die keine Tabellenattribute oder Datentypen definiert, um schemalosen und offenen Inhalt zu ermöglichen
Erstellen von Indizes	`CREATE INDEX`-Anweisung	`CREATE INDEX`Anweisung für alle Felder der obersten Ebene in einer Tabelle

Operation	RDBMS	QLDB
Einfügen von Daten	`INSERT`-Anweisung, die Werte innerhalb von neuen Zeilen oder Tupeln angibt, die dem von der Tabelle definierten Schema entsprechen.	`INSERT`-Anweisung, die Werte in einem neuen Dokument in einem gültigen Amazon Ion-Format angibt, unabhängig von den vorhandenen Dokumenten in der Tabelle
Abfragen von Daten	`SELECT-FROM-WHERE` -Anweisung	`SELECT-FROM-WHERE` -Anweisung in der gleichen Syntax wie SQL beim Abfragen von flachen Dokumenten
Aktualisieren von Daten	`UPDATE-SET-WHERE` -Anweisung	`UPDATE-SET-WHERE` -Anweisung in der gleichen Syntax wie SQL beim Aktualisieren von flachen Dokumenten
Löschen von Daten	`DELETE-FROM-WHERE` -Anweisung	`DELETE-FROM-WHERE` -Anweisung in der gleichen Syntax wie SQL beim Löschen von flachen Dokumenten
Verschachtelte und halbstrukturierte Daten	Nur flache Zeilen oder Tupel	Dokumente, die strukturierte, halbstrukturierte oder verschachtelte Daten umfassen können, die vom Amazon Ion-Datenformat und der PartiQL-Abfragesprache unterstützt werden
Abfragen von Metadaten	Keine integrierten Metadaten	`SELECT`-Anweisung, die Abfragen aus der integrierten Committed-Ansicht einer Tabelle ausführt

Operation	RDBMS	QLDB
Abfragen des Revisions verlaufs	Kein integrierter Datenverlauf	`SELECT`-Anweisung, die Abfragen aus der integrierten Verlaufsfunktion ausführt
Kryptografische Verifizierung	Keine integrierte Kryptogra phie oder Unveränderlichkeit	APIs, die einen Digest eines Journals und einen Nachweis zurückgeben, der die Integrität jeder Dokumentversion in Bezug auf diesen Digest überprüft

Einen Überblick über die Kernkonzepte und Terminologie in QLDB finden Sie unterSchlüsselkonzepte.

Ausführliche Informationen zum Erstellen, Abfragen und Verwalten von Daten in einem Ledger finden Sie unter Arbeiten mit Daten und Historie.

Kernkonzepte und Terminologie in Amazon QLDB

Dieser Abschnitt bietet einen Überblick über die wichtigsten Konzepte und Terminologie in Amazon QLDB, einschließlich der Buchstruktur und der Datenverwaltung in einem Ledger. Als Ledger-Datenbank unterscheidet sich QLDB von anderen dokumentenorientierten Datenbanken, wenn es um die folgenden Schlüsselkonzepte geht.

Themen

QLDB-Datenobjektmodell

Das grundlegende Datenobjektmodell in Amazon QLDB wird wie folgt beschrieben:

1. Hauptbuch

   Ihr erster Schritt besteht darin, ein Ledger zu erstellen. Dies ist der primäreAWS Ressourcentyp in QLDB. Informationen zum Erstellen eines Ledgers finden SieSchritt 1: Erstellen eines neuen Ledgers unter Erste Schritte mit der Konsole oderGrundfunktionen für Amazon QLDB-Ledgers.

   Sowohl für den`ALLOW_ALL` Modus als auch für den`STANDARD` Berechtigungsmodus eines Ledgers erstellen SieAWS Identity and Access Management (IAM) -Richtlinien, die Berechtigungen zum Ausführen von API-Vorgängen auf dieser Ledger-Ressource gewähren.

   Format des Ledger ARN:

   ```
   arn:aws:qldb:${region}:${account-id}:ledger/${ledger-name}
   ```

2. Journal und Tabellen

   Um mit dem Schreiben von Daten in ein QLDB-Ledger zu beginnen, erstellen Sie zunächst eine Tabelle mit einer grundlegendenCREATE TABLE Anweisung. Hauptbuchdaten bestehen aus Überarbeitungen von Dokumenten, die in das Journal des Ledgers aufgenommen wurden. Sie schreiben Änderungen an Dokumenten im Kontext von benutzerdefinierten Tabellen in das Ledger ein. In QLDB stellt eine Tabelle eine materialisierte Ansicht einer Sammlung von Dokumentrevisionen aus dem Journal dar.

   Im`STANDARD` Berechtigungsmodus eines Ledgers müssen Sie IAM-Richtlinien erstellen, die Berechtigungen zum Ausführen von PartiQL-Anweisungen auf dieser Tabellenressource gewähren. Mit Berechtigungen für eine Tabellenressource können Sie Anweisungen ausführen, die auf den aktuellen Status der Tabelle zugreifen. Sie können den Revisionsverlauf der Tabelle auch mithilfe der integriertenhistory() Funktion abfragen.

   Format des Tabellen-ARN:

   ```
   arn:aws:qldb:${region}:${account-id}:ledger/${ledger-name}/table/${table-id}
   ```

   Weitere Informationen zum Erteilen von Berechtigungen für ein Ledger und die zugehörigen Ressourcen finden Sie unterFunktionsweise von Amazon QLDB mit IAM.

3. Dokumente

   Tabellen bestehen aus Revisionen vonQLDB-Dokumente, bei denen es sich um Datensätze im Amazonstruct Ion-Format handelt. Eine Dokumentrevision stellt eine einzelne Version einer Sequenz von Dokumenten dar, die durch eine eindeutige Dokument-ID identifiziert werden.

   QLDB speichert die vollständige Änderungshistorie Ihrer übergebenen Dokumente. Mit einer Tabelle können Sie den aktuellen Status ihrer Dokumente abfragen, während Sie mit derhistory() Funktion den gesamten Revisionsverlauf der Dokumente einer Tabelle abfragen können. Einzelheiten zum Abfragen und Schreiben von Revisionen finden Sie unterArbeiten mit Daten und Historie.

4. Systemkatalog

   Jedes Ledger bietet auch eine systemdefinierte Katalogressource, die Sie abfragen können, um alle Tabellen und Indizes in einem Ledger aufzulisten. ImSTANDARD Berechtigungsmodus eines Ledgers benötigen Sie dieqldb:PartiQLSelect Berechtigung für diese Katalogressource, um Folgendes zu tun:

   - Führen SieSELECT Anweisungen für die Systemkatalogtabelle information_schema.user_tables aus.
   - Sehen Sie sich Tabellen- und Indexinformationen auf der Ledger-Detailseite der QLDB-Konsole an.
   - Sehen Sie sich die Liste der Tabellen und Indizes im PartiQL-Editor auf der QLDB-Konsole an.

   Format des Katalog-ARN:

   ```
   arn:aws:qldb:${region}:${account-id}:ledger/${ledger-name}/information_schema/
   user_tables
   ```

Transaktionen, bei denen das Journal im Vordergrund steht

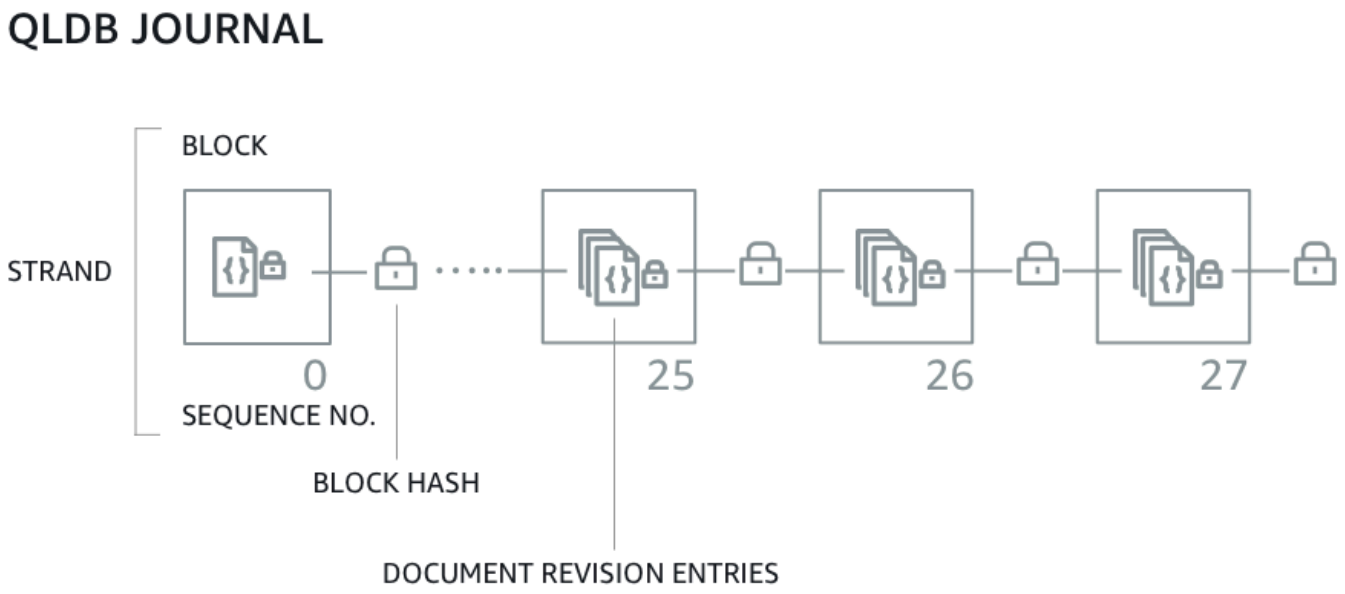
Wenn eine Anwendung Daten in einem QLDB-Ledger liest oder schreibt, tut sie dies in einer Datenbanktransaktion. Alle Transaktionen unterliegen den in definierten LimitsKontingente und Limits in Amazon QLDB. Innerhalb einer Transaktion führt QLDB die folgenden Schritte aus:

1. Lesen Sie den aktuellen Status der Daten aus dem Ledger.

2. Führen Sie die in der Transaktion angegebenen Anweisungen aus und überprüfen Sie dann mithilfe von Optimistic Concurrency Control (OCC), ob Konflikte vorliegen, um eine vollständig serialisierbare Isolierung sicherzustellen.
3. Wenn keine OCC-Konflikte gefunden werden, geben Sie die Transaktionsergebnisse wie folgt zurück:
   - Geben Sie für Lesevorgänge die Ergebnismenge zurück und übergeben Sie die`SELECT` Anweisungen ausschließlich als Anhängen an das Journal.
   - Übertragen Sie beim Schreiben alle Aktualisierungen, Löschungen oder neu eingefügten Daten ausschließlich durch Anhängen in das Journal.

Das Journal stellt eine vollständige und unveränderliche Historie aller Änderungen an Ihren Daten dar. QLDB schreibt in einer Transaktion einen verketteten Block in das Journal. Jeder Block enthält Eingabeobjekte, die die Dokumentrevisionen darstellen, die Sie einfügen, aktualisieren und löschen, zusammen mit den PartiQL-Anweisungen, durch die sie festgeschrieben wurden.

Das folgende Diagramm veranschaulicht diese Journalstruktur.



Das Diagramm zeigt, dass Transaktionen in das Journal als Blöcke übergeben werden, die Dokumentrevisionseinträge enthalten. Jeder Block wird gehasht und zur Überprüfung mit den nachfolgenden Blöcken verkettet. Jeder Block verfügt über eine Sequenznummer, um seine Adresse in der Strähne anzugeben.

> **Note**
>
> In Amazon QLDB ist ein Strang eine Partition des Journals Ihres Ledgers. QLDB unterstützt derzeit nur Zeitschriften mit einem einzigen Strang.

Hinweise zum Dateninhalt in einem Block finden Sie unter Journalinhalte in Amazon QLDB.

Abfrage Ihrer Daten

QLDB ist für Workloads bei der Online-Transaktionsverarbeitung (OLTP) konzipiert. Ein Ledger bietet abfragbare Tabellenansichten Ihrer Daten auf der Grundlage der Transaktionsinformationen, die in das Journal übernommen wurden. Eine Tabellenansicht in QLDB ist eine Teilmenge der Daten in einer Tabelle. Ansichten werden in Echtzeit verwaltet, sodass sie immer verfügbar für abzufragende Anwendungen sind.

Sie können die folgenden systemdefinierten Ansichten mithilfe von`SELECT` PartiQL-Anweisungen abfragen:

- Benutzer — Die letzte aktive Revision nur der Daten, die Sie in die Tabelle geschrieben haben (d. h. der aktuelle Status Ihrer Benutzerdaten). Dies ist die Standardansicht in QLDB.
- Committed — Die letzte aktive Revision sowohl Ihrer Benutzerdaten als auch der systemgenerierten Metadaten. Dies ist die vollständige systemdefinierte Tabelle, die direkt Ihrer Benutzertabelle entspricht.

Zusätzlich zu diesen abfragbaren Ansichten können Sie den Revisionsverlauf Ihrer Daten mithilfe der integrierten Funktion abfragenVerlaufsfunktion. Die Verlaufsfunktion gibt sowohl Ihre Benutzerdaten als auch die zugehörigen Metadaten im gleichen Schema wie die Committed-Ansicht zurück.

Datenspeicherung

Es gibt zwei Arten von Datenspeicherungsarten in QLDB:

- Journalspeicher — Der Festplattenspeicher, der vom Journal eines Ledgers belegt wird. Das Journal kann nur angehängt werden (Append-only) und enthält den vollständigen, unveränderlichen und überprüfbaren Verlauf aller Datenänderungen.

- Indizierter Speicher — Der Festplattenspeicher, der von den Tabellen, Indizes und der indizierten Historie eines Ledgers verwendet wird. Der indizierte Speicher besteht aus für Hochleistungs-Abfragen optimierten Ledgerdaten.

Nachdem Ihre Daten in das Journal übernommen wurden, werden sie in den von Ihnen definierten Tabellen materialisiert. Diese Tabellen sind für schnellere und effizientere Abfragen optimiert. Wenn eine Anwendung die Transaktionsdaten-API zum Lesen von Daten verwendet, greift sie auf die Tabellen und Indizes zu, die in Ihrem indizierten Speicher gespeichert sind.

Modell des QLDB API ARN ARN

QLDB bietet zwei Arten von APIs, mit denen Ihr Anwendungscode interagieren kann:

- Amazon QLDB — Die QLDB-Ressourcenmanagement-API (auch als Steuerungsebene bekannt). Diese API wird nur für die Verwaltung von Ledger-Ressourcen und für nicht transaktionale Datenoperationen verwendet. Sie können diese Operationen verwenden, um Ledger zu erstellen, zu löschen, zu beschreiben, aufzulisten und zu aktualisieren. Sie können Daten auch kryptografisch überprüfen und Journalblöcke exportieren oder streamen.
- Amazon QLDB-Sitzung — Die QLDB-Transaktionsdaten-API. Sie können diese API verwenden, um Datentransaktionen in einem Ledger mit PartiQL-Anweisungen auszuführen.

> **Important**
>
> Anstatt direkt mit der QLDB Session API zu interagieren, empfehlen wir, den QLDB-Treiber oder die QLDB-Shell zu verwenden, um Datentransaktionen auf einem Ledger auszuführen.
>
> - Wenn Sie mit einemAWS SDK arbeiten, verwenden Sie den QLDB-Treiber. Der Treiber stellt eine abstrakte Ebene über der QLDB-Sitzungsdaten-API bereit und verwaltet den`SendCommand` Vorgang für Sie. Für weitere Informationen und eine Liste der unterstützten Programmiersprachen siehe Erste Schritte mit dem Treiber.
> - Wenn Sie mit der arbeitenAWS CLI, verwenden Sie die QLDB-Shell. Die Shell ist eine Befehlszeilenschnittstelle, die den QLDB-Treiber verwendet, um mit einem Ledger zu interagieren. Weitere Informationen finden Sie unter Verwenden der Amazon QLDB-Shell (nur Daten-API).

Weitere Informationen zu diesen APIOperationen finden Sie imAmazon QLDB API-Referenz.

Nächste Schritte

Informationen zur Verwendung eines Ledgers mit Ihren Daten finden Sie in den Beispielen, in denen das Erstellen von Tabellen, das Einfügen von DatenArbeiten mit Daten und Historie in Amazon QLDB und das Ausführen einfacher Abfragen beschrieben wird, und folgen Sie ihnen. In diesem Handbuch wird anhand von Beispieldaten und Abfragebeispielen eingehend erläutert, wie diese Konzepte funktionieren.

Einen schnellen Einstieg mit einem Tutorial zur Beispielanwendung mit der QLDB-Konsole finden Sie unterErste Schritte mit der Amazon QLDB-Konsole.

Eine Liste der Schlüsselbegriffe und Definitionen, die in diesem Abschnitt beschrieben werden, finden Sie unter Amazon QLDB Glossar.

Journalinhalte in Amazon QLDB

In Amazon QLDB ist das Journal das unveränderliche Transaktionsprotokoll, in dem der vollständige und überprüfbare Verlauf aller Änderungen an Ihren Daten gespeichert wird. Das Journal ist nur für Anhänge bestimmt und besteht aus einer sequenzierten und mit Hash-Verkettung versehenen Gruppe von Blöcken, die Ihre übermittelten Daten und andere Systemmetadaten enthalten. QLDB schreibt in einer Transaktion einen verketteten Block in das Journal.

Dieser Abschnitt enthält ein Beispiel für einen Journalblock mit Beispieldaten und beschreibt den Inhalt eines Blocks.

Themen

Blockbeispiel

Ein Journalblock enthält Transaktionsmetadaten zusammen mit Einträgen, die die Dokumentrevisionen darstellen, die in der Transaktion festgeschrieben wurden, sowie die PartiQL-Anweisungen, die diese festgeschrieben haben.

Im Folgenden finden Sie ein Beispiel für einen Block mit Beispieldaten.

> **Note**
>
> Dieses Blockbeispiel dient nur zu Informationszwecken. Die angezeigten Hashes sind keine echten berechneten Hashwerte.

```
{
  blockAddress:{
    strandId:"4o5UuzWSW5PIoOGm5jPA6J",
    sequenceNo:25
  },
  transactionId:"3gtB8Q8dfIMA8lQ5pzHAMo",
  blockTimestamp:2022-06-08T18:46:46.512Z,
  blockHash:{{QS5lJt8vRxT30L9OGL5oU1pxFTe+UlEwakYBCrvGQ4A=}},
  entriesHash:{{buYYc5kV4rrRtJAsrIQnfnhgkzfQ8BKjI0C2vFnYQEw=}},
  previousBlockHash:{{I1UKRIWUgkM1X6042kcoZ/eN1rn0uxhDTc08zw9kZ5I=}},
  entriesHashList:[
    {{BUCXP6oYgmug2AfPZcAZup2lKolJNTbTuV5RA1VaFpo=}},
    {{cTIRkjuULzp/4KaUEsb/S7+TG8FvpFiZHT4tEJGcANc=}},
    {{3aktJSMyJ3C5StZv4WIJLu/w3D8mGtduZvP0ldKUaUM=}},
    {{GPKIJ1+o8mMZmPj/35ZQXoca2z64MVYMCwqs/g080IM=}}
  ],
  transactionInfo:{
    statements:[
      {
        statement:"INSERT INTO VehicleRegistration VALUE ?",
        startTime:2022-06-08T18:46:46.063Z,
        statementDigest:{{KY2nL6UGUPs5lXCLVXcUaBxcEIop0Jvk4MEjcFVBfwI=}}
      },
      {
        statement:"SELECT p_id FROM Person p BY p_id WHERE p.FirstName = ? and
 p.LastName = ?",
        startTime:2022-06-08T18:46:46.173Z,
        statementDigest:{{QS2nfB8XBf2ozlDx0nvtsliOYDSmNHMYC3IRH4Uh690=}}
      },
      {
        statement:"UPDATE VehicleRegistration r SET r.Owners.PrimaryOwner.PersonId = ?
 WHERE r.VIN = ?",
        startTime:2022-06-08T18:46:46.278Z,
        statementDigest:{{nGtIA9Qh0/dwIplOR8J5CTeqyUVtNUQgXfltDUo2Aq4=}}
```

```
      },
      {
        statement:"DELETE FROM DriversLicense l WHERE l.LicenseNumber = ?",
        startTime:2022-06-08T18:46:46.385Z,
        statementDigest:{{ka783dcEP58Q9AVQ1m9NOJd3JAmEvXLjzl0OjN1BojQ=}}
      }
    ],
    documents:{
      HwVFkn8IMRa0xjze5xcgga:{
        tableName:"VehicleRegistration",
        tableId:"HQZ6cgIMUi204Lq1tT4oaJ",
        statements:[0,2]
      },
      IiPTRxLGJZa342zHFCFT15:{
        tableName:"DriversLicense",
        tableId:"BvtXEB1JxZg0lJlBAtbtSV",
        statements:[3]
      }
    }
  },
  revisions:[
    {
      hash:{{FR1IWcWew0yw1TnRklo2YMF/qtwb7ohsu5FD8A4DSVg=}}
    },
    {
      blockAddress:{
        strandId:"4o5UuzWSW5PIoOGm5jPA6J",
        sequenceNo:25
      },
      hash:{{6TTHbcfIVdWoFC/j90BOZi0JdHzhjSXo1tW+uHd6Dj4=}},
      data:{
        VIN:"1N4AL11D75C109151",
        LicensePlateNumber:"LEWISR261LL",
        State:"WA",
        City:"Seattle",
        PendingPenaltyTicketAmount:90.25,
        ValidFromDate:2017-08-21,
        ValidToDate:2020-05-11,
        Owners:{
          PrimaryOwner:{
            PersonId:"3Ax20JIix5J2ulu2rCMvo2"
          },
          SecondaryOwners:[]
        }
```

```
      },
      metadata:{
        id:"HwVFkn8IMRa0xjze5xcgga",
        version:0,
        txTime:2022-06-08T18:46:46.492Z,
        txId:"3gtB8Q8dfIMA8lQ5pzHAMo"
      }
    },
    {
      blockAddress:{
        strandId:"4o5UuzWSW5PIoOGm5jPA6J",
        sequenceNo:25
      },
      hash:{{ZVF/f1uSqd5DIMqzI04CCHaCGFK/J0Jf5AFzSEk0l90=}},
      metadata:{
        id:"IiPTRxLGJZa342zHFCFT15",
        version:1,
        txTime:2022-06-08T18:46:46.492Z,
        txId:"3gtB8Q8dfIMA8lQ5pzHAMo"
      }
    }
  ]
}
```

Im `revisions`-Feld enthalten einige Revisionsobjekte möglicherweise nur einen `hash`-Wert und keine anderen Attribute. Dabei handelt es sich um interne Systemrevisionen, die keine Benutzerdaten enthalten. Die Hashes dieser Revisionen sind Teil der vollständigen Hashkette des Journals, die für die kryptografische Überprüfung erforderlich ist.

Blockinhalte

Ein Journalblock verfügt über die folgenden Felder:

`blockAddress`

Die Position des Blocks im Journal. Ein Adresse ist eine Amazon Ion-Struktur, die über zwei Felder verfügt: `strandId` und `sequenceNo`.

Beispiel: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

`transactionId`

Die eindeutige ID der Transaktion, die die den Block festgeschrieben hat.

`blockTimestamp`

Der Zeitstempel, als der Block im Journal festgeschrieben wurde.

`blockHash`

Der 256-Bit-Hash-Wert, der den Block eindeutig darstellt. Dies ist der Hash der Verkettung von `entriesHash` und `previousBlockHash`.

`entriesHash`

Der Hash, der alle Einträge innerhalb des Blocks darstellt, einschließlich ausschließlich interner Systemeinträge. Dies ist der Wurzelhash des Merkle-Baums, in dem die Blattknoten aus allen Hashes in `entriesHashList` bestehen.

`previousBlockHash`

Der Hash des vorherigen verketteten Blocks im Journal.

`entriesHashList`

Die Liste der Hashes, die jeden Eintrag innerhalb des Blocks darstellen. Diese Liste kann die folgenden Eintrags-Hashes enthalten:

- Der Ionen-Hash, der darstellt`transactionInfo`. Dieser Wert wird berechnet, indem der Ionen-Hash der gesamten`transactionInfo` Struktur genommen wird.
- Der Wurzelhash des Merkle-Baums, in dem die Blattknoten aus allen Hashes in `revisions` bestehen.
- Der Ionen-Hash, der darstellt`redactionInfo`. Dieser Hash existiert nur in Blöcken, die durch eine Redaktionstransaktion festgeschrieben wurden. Sein Wert wird berechnet, indem der Ionen-Hash der gesamten`redactionInfo` Struktur genommen wird.
- Hashes, die nur interne Systemmetadaten darstellen. Diese Hashes sind möglicherweise nicht in allen Blöcken vorhanden.

`transactionInfo`

Eine Amazon Ion-Struktur, die Informationen zu den Anweisungen in der Transaktion enthält, die den Block festgeschrieben hat. Diese Struktur enthält die folgenden Felder:

- `statements`— Die Liste der PartiQL-Anweisungen und der`startTime` Zeitpunkt, zu dem sie ausgeführt wurden. Jede Anweisung hat einen `statementDigest`-Hash, der benötigt wird, um den Hash der `transactionInfo`-Struktur zu berechnen.

- `documents`— Die Dokument-IDs, die durch die Kontoauszüge aktualisiert wurden. Jedes Dokument enthält`tableId` das`tableName` und zu dem es gehört, und den Index jeder Anweisung, mit der es aktualisiert wurde.

`revisions`

Die Liste der Dokumentversionen, die im Block festgeschrieben wurden. Jede Revisionsstruktur enthält alle Felder aus der bestätigten Ansicht der Revision.

Hierzu können auch Hashes gehören, die rein interne Systemrevisionen darstellen, die Teil der vollständigen Hashkette eines Journals sind.

Redigierte Versionen

In Amazon QLDB löscht eine`DELETE` Anweisung ein Dokument nur logisch, indem eine neue Revision erstellt wird, die es als gelöscht markiert. QLDB unterstützt auch einen Datenredigierungsvorgang, mit dem Sie inaktive Dokumentrevisionen in der Historie einer Tabelle dauerhaft löschen können.

Beim Schwärzen werden nur die Benutzerdaten in der angegebenen Revision gelöscht, und die Journalsequenz und die Metadaten des Dokuments bleiben unverändert. Dadurch bleibt die allgemeine Datenintegrität Ihres Ledgers erhalten. Weitere Informationen und ein Beispiel für einen Redigierungsvorgang finden Sie unterRedigieren von Dokumentrevisionen.

Beispiel für Revision

Betrachten Sie das vorherige Blockbeispiel. Nehmen Sie in diesem Block an, dass Sie die Revision redigieren, die eine Dokument-ID von`HwVFkn8IMRa0xjze5xcgga` und eine Versionsnummer von hat`0`.

Nach Abschluss der Bearbeitung werden die Benutzerdaten in der Revision (dargestellt durch die`data` Struktur) durch ein neues`dataHash` Feld ersetzt. Der Wert dieses Feldes ist der Ionen-Hash der entfernten`data` Struktur. Dadurch behält das Ledger seine allgemeine Datenintegrität bei und bleibt durch die vorhandenen Verifizierungs-API-Operationen kryptografisch überprüfbar.

Das folgende Revisionsbeispiel zeigt die Ergebnisse dieser Schwärzung, wobei das neue`dataHash` Feld *`rot kursiv`* hervorgehoben ist.

> **Note**
>
> Dieses Revisionsbeispiel dient nur zu Informationszwecken. Die angezeigten Hashes sind keine echten berechneten Hashwerte.

```
...
{
  blockAddress:{
    strandId:"4o5UuzWSW5PIoOGm5jPA6J",
    sequenceNo:25
  },
  hash:{{6TTHbcfIVdWoFC/j90BOZi0JdHzhjSXo1tW+uHd6Dj4=}},
  dataHash:{{s83jd7sfhsdfhksj7hskjdfjfpIPP/DP2hvionas2d4=}},
  metadata:{
    id:"HwVFkn8IMRa0xjze5xcgga",
    version:0,
    txTime:2022-06-08T18:46:46.492Z,
    txId:"3gtB8Q8dfIMA8lQ5pzHAMo"
  }
}
...
```

QLDB fügt dem Journal außerdem einen neuen Block für die abgeschlossene Redaktionsanfrage an. Dieser Block enthält einen zusätzlichen`redactionInfo` Eintrag, der eine Liste der Revisionen enthält, die in der Transaktion redigiert wurden, wie im folgenden Beispiel gezeigt.

```
...
redactionInfo:{
  revisions:[
    {
      blockAddress:{
        strandId:"4o5UuzWSW5PIoOGm5jPA6J",
        sequenceNo:25
      },
      tableId:"HQZ6cgIMUi204Lq1tT4oaJ",
      documentId:"HwVFkn8IMRa0xjze5xcgga",
      version:0
    }
  ]
}
```

```
...
```

Beispielanwendung

Ein Java-Codebeispiel, das die Hashkette eines Journals anhand exportierter Daten validiert, finden Sie im GitHub Repository aws-samples/amazon-qldb-dmv-sample -java. Diese Beispielanwendung enthält die folgenden Klassendateien:

- ValidateQldbHashChain.java — Enthält Tutorialcode, der Journalblöcke aus einem Ledger exportiert und die exportierten Daten verwendet, um die Hash-Kette zwischen Blöcken zu validieren.
- JournalBlock.java — Enthält eine Methode mit dem Namen`verifyBlockHash()`, die demonstriert, wie jede einzelne Hash-Komponente innerhalb eines Blocks berechnet wird. Diese Methode wird durch den Tutorialcode in aufgerufen`ValidateQldbHashChain.java`.

Anweisungen zum Herunterladen und Installieren dieser vollständigen Beispielanwendung finden Sie unter Installation der Amazon QLDB Java-Beispielanwendung. Bevor Sie den Tutorialcode ausführen, stellen Sie sicher, dass Sie die Schritte 1—3 imJava-Anleitung Abschnitt So richten Sie ein Beispielbuch ein und laden Sie es mit Beispieldaten.

Weitere Informationen finden Sie auch unter

Weitere Informationen zu -Journals in QLDB finden Sie in den folgenden Themen:

- Exportieren von Journaldaten aus Amazon QLDB— Um zu erfahren, wie Journaldaten in den Amazon Simple Storage Service (Amazon S3) exportiert werden.
- Streaming von Journaldaten aus Amazon QLDB— Um zu erfahren, wie Journaldaten an Amazon Kinesis Data Streams gestreamt werden.
- Datenverifizierung in Amazon QLDB— Um mehr über die kryptografische Überprüfung von Journaldaten zu erfahren.

Amazon QLDB Glossar

Im Folgenden finden Sie Definitionen für wichtige Begriffe, die Ihnen bei der Arbeit mit Amazon QLDB begegnen könnten.

Block | Digest | document | Dokument-ID | Dokumentrevision | Eintrag | field | index | Indizierter Speicher | Journal | Journalblock | journal storage | Journal-Strähne | Journaltipp | Ledger | Nachweis | Änderung | Sitzung | Strang | Tabelle | tabellarische Ansicht | Ansicht

Block

Ein Objekt, das in einem Journal in einer Transaktion festgeschrieben wird. Eine einzelne Transaktion schreibt einen Block in das Journal, sodass ein Block nur einer Transaktion zugeordnet werden kann. Ein Block enthält Einträge, die die Dokumentrevisionen darstellen, die in der Transaktion bestätigt wurden, zusammen mit den PartiQL-Anweisungen, die sie bestätigt haben.

Jeder Block enthält außerdem einen Hash-Wert für die Verifizierung. Ein Block-Hash wird aus den Eintragshashes innerhalb dieses Blocks zusammen mit dem Hash des vorherigen verketteten Blocks berechnet.

Digest

Ein 256-Bit-Hashwert, der den gesamten Verlauf der Dokumentenrevisionen Ihres Hauptbuchs zu einem bestimmten Zeitpunkt eindeutig darstellt. Ein Digest-Hash wird aus der vollständigen Hash-Kette Ihres Journals ab dem letzten festgeschriebenen Block im Journal zu diesem Zeitpunkt berechnet.

Mit QLDB können Sie einen Digest als sichere Ausgabedatei generieren. Anschließend können Sie diese Ausgabedatei verwenden, um die Integrität Ihrer Dokumentrevisionen in Bezug auf diesen Hash zu verifizieren.

document

Ein Datensatz im Amazon Ion-Format `struct` kann eingefügt, aktualisiert und in einer Tabelle gelöscht werden. Ein QLDB-Dokument kann strukturierte, halbstrukturierte, verschachtelte und schemalose Daten enthalten.

Dokument-ID

Die UUID (Universally Unique Identifier), die QLDB jedem Dokument zuweist, das Sie in eine Tabelle einfügen. Diese ID ist eine 128-Bit-Zahl, die in einer Base62-kodierten alphanumerischen Zeichenfolge mit einer festen Länge von 22 Zeichen dargestellt wird.

Dokumentrevision

Eine Ionen-Struktur, die eine einzelne Version einer Sequenz von Dokumenten darstellt, die durch eine eindeutige Dokument-ID identifiziert werden. Eine Revision umfasst sowohl

Ihre Benutzerdaten (d. h. die Daten, die Sie in die Tabelle geschrieben haben) als auch systemgenerierte Metadaten. Jede Revision ist einer Tabelle zugeordnet und wird durch eine Kombination aus der Dokument-ID und einer auf Null basierenden Versionsnummer eindeutig identifiziert.

Eintrag

Ein Objekt, das in einem Block enthalten ist. Einträge repräsentieren Dokumentrevisionen, die in einer Transaktion eingefügt, aktualisiert und gelöscht werden, zusammen mit den PartiQL-Anweisungen, die diese festschreiben.

Jeder Eintrag enthält außerdem einen Hash-Wert für die Verifizierung. Ein Eintragshash wird aus den Revisionshashes oder den Anweisungshashes innerhalb dieses Eintrags berechnet.

field

Ein Name-Wert-Paar, aus dem jedes Attribut eines QLDB-Dokuments besteht. Der Name ist ein Symbol-Token, und der Wert ist uneingeschränkt.

index

Eine Datenstruktur, die Sie für eine Tabelle erstellen können, um die Leistung von Datenabrufvorgängen zu optimieren. Informationen zu Indizes in QLDB finden SieCREATE INDEX in der Amazon QLDB PartiQL-Referenz.

Indizierter Speicher

Der Speicherplatz, der von den Tabellen, Indizes und vom indizierten Verlauf eines Ledgers verwendet wird. Der indizierte Speicher besteht aus für Hochleistungs-Abfragen optimierten Ledgerdaten.

Journal

Die Hash-verkettete Gruppe aller Blöcke, die in Ihrem Ledger übergeben werden. Das Journal kann nur angehängt werden (Append-only) und stellt einen vollständigen und unveränderlichen Verlauf aller Änderungen an Ihren Ledger-Daten dar.

Journalblock

Siehe Block.

journal storage

Der Speicherplatz, der vom Journal eines Ledgers verwendet wird.

Journal-Strähne

Siehe Strang.

Journaltipp

Der letzte festgeschriebene Block in einem Journal zu einem bestimmten Zeitpunkt.

Ledger

Eine Instanz einer Amazon QLDB-Ledger-Datenbankressource. Dies ist der primäreAWS Ressourcentyp in QLDB. Ein Ledger besteht sowohl aus dem Journal-Speicher als auch aus dem indizierten Speicher. Nachdem die Buchdaten in das Journal übernommen wurden, können sie in Tabellen mit Revisionen von Amazon Ion-Dokumenten abgefragt werden.

Nachweis

Die geordnete Liste von 256-Bit-Hashwerten, die QLDB für eine bestimmte Zusammenfassung und Dokumentrevision zurückgibt. Sie besteht aus den Hashes, die von einem Hash-Baummodell benötigt werden, um den gegebenen Revisions-Hash mit dem Digest-Hash zu verketten. Sie verwenden einen Nachweis, um die Integrität Ihrer Änderungen im Vergleich zum Digest zu überprüfen. Weitere Informationen finden Sie unter Datenverifizierung in Amazon QLDB.

Änderung

Siehe Dokumentrevision.

Sitzung

Ein Objekt, das Informationen über Ihre Datentransaktionsanfragen und Antworten an und aus einem Ledger verwaltet. Eine aktive Sitzung (eine, die aktiv eine Transaktion ausführt) stellt eine einzelne Verbindung zu einem Ledger dar. QLDB unterstützt eine aktiv laufende Transaktion pro Sitzung.

Strang

Eine Partition eines Journals. QLDB unterstützt derzeit nur Zeitschriften mit einem einzigen Strang.

Tabelle

Eine materialisierte Ansicht einer ungeordneten Sammlung von Dokumentenüberarbeitungen, die im Journal des Hauptbuchs aufgezeichnet wurden.

tabellarische Ansicht

Eine abfragbare Teilmenge der Daten in einer Tabelle, die auf Transaktionen basiert, die an das Journal übertragen wurden. In einer PartiQL-Anweisung wird eine Ansicht durch ein Präfixkennzeichen (beginnend mit`_ql_`) für einen Tabellennamen gekennzeichnet.

Sie können die folgenden systemdefinierten Ansichten mithilfe von`SELECT` Anweisungen abfragen:

- Benutzer — Die letzte aktive Revision nur der Daten, die Sie in die Tabelle geschrieben haben (d. h. der aktuelle Status Ihrer Benutzerdaten). Dies ist die Standardansicht in QLDB.
- Committed — Die letzte aktive Revision sowohl Ihrer Benutzerdaten als auch der vom System generierten Metadaten. Dies ist die vollständige systemdefinierte Tabelle, die direkt Ihrer Benutzertabelle entspricht. Zum Beispiel: `_ql_committed_`*`TableName`*.

Ansicht

Siehe tabellarische Ansicht.

Zugreifen auf Amazon QLDB

Sie können mit der AWS Command Line Interface (AWS CLI) oder der AWS Management Console QLDB-API auf Amazon QLDB zugreifen. In den folgenden Abschnitten werden die Verwendung dieser Optionen und die Voraussetzungen für ihre Verwendung beschrieben.

Voraussetzungen

Bevor Sie auf QLDB zugreifen können, müssen Sie eine einrichten, AWS-Konto falls Sie dies noch nicht getan haben.

Themen

So melden Sie sich für ein AWS-Konto an

Wenn Sie kein AWS-Konto haben, führen Sie die folgenden Schritte zum Erstellen durch.

Anmeldung für ein AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für ein AWS-Konto anmelden, wird ein Root-Benutzer des AWS-Kontos erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen eine Bestätigungs-E-Mail, sobald die Anmeldung abgeschlossen ist. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Einen Administratorbenutzer erstellen

Nachdem Sie sich für einen angemeldet habenAWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-KontosAWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Schützen Ihres Root-Benutzer des AWS-Kontos

1. Melden Sie sich bei AWS Management Console als Kontobesitzer an, indem Sie Stammbenutzer auswählen und Ihre AWS-Konto-E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-AnmeldungBenutzerhandbuch zu .
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen dazu finden Sie unter Aktivieren eines virtuellen MFA-Geräts für den Root-Benutzer Ihres AWS-Konto (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Administratorbenutzers

1. Aktivieren Sie IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity CenterBenutzerhandbuch.
2. Gewähren Sie in IAM Identity Center einem Administratorbenutzer Administratorzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit der Standardeinstellung konfigurieren IAM-Identity-Center-Verzeichnis im AWS IAM Identity CenterBenutzerhandbuch.

Als Administratorbenutzer anmelden

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM-Identity-Center-Benutzer finden Sie unter Anmelden beim AWS-Zugangsportal im AWS-Anmeldung Benutzerhandbuch zu.

QLDB-Berechtigungen in IAM verwalten

Hinweise zur Verwendung von AWS Identity and Access Management (IAM) zur Verwaltung von QLDB-Berechtigungen für Benutzer finden Sie unter. Funktionsweise von Amazon QLDB mit IAM

Gewähren Sie programmatischen Zugriff (optional)

Benutzer benötigen programmgesteuerten Zugriff, wenn sie außerhalb der AWS Management Console mit AWS interagieren möchten. Die Vorgehensweise, um programmgesteuerten Zugriff zu gewähren, hängt davon ab, welcher Benutzertyp auf zugreift AWS.

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Konfigurieren der AWS CLI für die Verwendung von AWS IAM Identity Center im AWS Command Line Interface-Benutzerhandbuch. • Informationen zu AWS-SDKs, Tools und AWS-APIs

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		finden Sie unter IAM-Identity-Center-Authentifizierung im Referenzhandbuch zu AWS-SDKs und Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS-Ressourcen im IAM-Benutzerhandbuch.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Authentifizierung mit IAM-Benutzer-Anmeldeinformationen im AWS Command Line Interface-Benutzerhandbuch. • Informationen zu AWS-SDKs und Tools finden Sie unter Authentifizierung mit langfristigen Anmeldeinformationen im Referenzhandbuch zu AWS-SDKs und Tools. • Informationen zu AWS-APIs finden Sie unter Verwalten von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch.

So greifen Sie auf Amazon QLDB zu

Nachdem Sie die Voraussetzungen für die Einrichtung eines erfüllt habenAWS-Konto, finden Sie in den folgenden Themen weitere Informationen zum Zugriff auf QLDB:

- Verwenden der Konsole
- Verwendung der AWS CLI (nur Verwaltungs-API)
- Verwenden der Amazon QLDB-Shell (nur Daten-API)

- Verwenden der API

Zugreifen auf Amazon QLDB über die Konsole

Sie können auf die QLDB AWS Management Console für Amazon unter https://console.aws.amazon.com/qldb zugreifen.

Sie können die Konsole verwenden, um in QLDB Folgendes zu tun:

- Erstellen, löschen, beschreiben und listen Sie Ledgers auf.
- Führen Sie PartiQL-Anweisungen mit dem PartiQL-Editor aus.
- Verwalten Sie Tags für QLDB-Ressourcen.
- Überprüfen Sie die Journaldaten kryptografisch.
- Exportieren oder Streamen von Journalblöcken.

Informationen zum Erstellen eines Amazon QLDB-Ledgers und dessen Einrichtung mit Beispielanwendungsdaten finden Sie unter. Erste Schritte mit der Amazon QLDB-Konsole

Kurzreferenz zum PartiQL-Editor

Amazon QLDB unterstützt eine Teilmenge von PartiQL als Abfragesprache und Amazon Ion als dokumentenorientiertes Datenformat. Eine vollständige Anleitung und detailliertere Informationen zur QLDB-Implementierung von PartiQL finden Sie in der. Amazon QLDB PartiQL-Referenz

Die folgenden Themen bieten einen kurzen Überblick über die Verwendung von PartiQL in QLDB.

Themen

- PartiQL Kurztipps in QLDB
- Befehle
- Systemdefinierte Ansichten
- Grundlegende Syntaxregeln
- Tastenkombinationen für den PartiQL-Editor

PartiQL Kurztipps in QLDB

Im Folgenden finden Sie eine kurze Zusammenfassung von Tipps und bewährten Methoden für die Arbeit mit PartiQL in QLDB:

- Machen Sie sich mit Parallelität und Transaktionslimits vertraut — Alle Anweisungen, einschließlich `SELECT` Abfragen, unterliegen Konflikten mit optimistischer Parallelitätskontrolle (OCC) und Transaktionslimits, einschließlich eines Transaktions-Timeouts von 30 Sekunden.
- Verwenden Sie Indizes — Verwenden Sie Indizes mit hoher Kardinalität und führen Sie gezielte Abfragen durch, um Ihre Anweisungen zu optimieren und vollständige Tabellenscans zu vermeiden. Weitere Informationen hierzu finden Sie unter Optimieren der Abfrageleistung.
- Verwenden Sie Gleichheitsprädikate — Indizierte Suchvorgänge erfordern einen Gleichheitsoperator (oder). `= IN` Ungleichheitsoperatoren (<,, `>LIKE,BETWEEN`) kommen nicht für indizierte Suchvorgänge in Frage und führen zu vollständigen Tabellenscans.
- Nur innere Joins verwenden — QLDB unterstützt nur innere Joins. Es hat sich bewährt, Felder miteinander zu verknüpfen, die für jede Tabelle, die Sie verknüpfen, indexiert sind. Wählen Sie Indizes mit hoher Kardinalität sowohl für die Verbindungskriterien als auch für die Gleichheitsprädikate.

Befehle

QLDB unterstützt die folgenden PartiQL-Befehle.

Data Definition Language (DDL)

Befehl	Beschreibung
CREATE INDEX	Erstellt einen Index für ein Dokumentfeld der obersten Ebene in einer Tabelle.
CREATE TABLE	Erstellt eine -Tabelle.
DROP INDEX	Löscht einen Index aus einer Tabelle.
DROP TABLE	Deaktiviert eine bestehende Tabelle.
UNDROP TABLE	Reaktiviert eine inaktive Tabelle.

Datenmanipulationssprache (DML)

Befehl	Beschreibung
DELETE	Markiert ein aktives Dokument als gelöscht, indem eine neue, endgültige Version des Dokuments erstellt wird.
VON (INSERT, REMOVE oder SET)	Semantisch dasselbe wieUPDATE.
INSERT	Fügt einer Tabelle ein oder mehrere Dokumente hinzu.
SELECT	Ruft Daten aus einer oder mehreren Tabellen ab.
UPDATE	Aktualisiert, fügt bestimmte Elemente in einem Dokument ein oder entfernt sie.

Beispiele für DML-Anweisungen

EINFÜGEN

```
INSERT INTO VehicleRegistration VALUE
{
    'VIN' : 'KM8SRDHF6EU074761', --string
    'RegNum' : 1722, --integer
    'PendingPenaltyTicketAmount' : 130.75, --decimal
    'Owners' : { --nested struct
        'PrimaryOwner' : { 'PersonId': '294jJ3YUoH1IEEm8GSabOs' },
        'SecondaryOwners' : [ --list of structs
            { 'PersonId' : '1nmeDdLo3AhGswBtyM1eYh' },
            { 'PersonId': 'IN7MvYtUjkp1GMZu0F6CG9' }
        ]
    },
    'ValidToDate' : `2020-06-25T` --Ion timestamp literal with day precision
}
```

AKTUALISIEREN—EINFÜGEN

```
UPDATE Vehicle AS v
```

```
INSERT INTO v VALUE 26500 AT 'Mileage'
WHERE v.VIN = '1N4AL11D75C109151'
```

AKTUALISIEREN—ENTFERNEN

```
UPDATE Person AS p
REMOVE p.Address
WHERE p.GovId = '111-22-3333'
```

SELECT — Korrelierte Unterabfrage

```
SELECT r.VIN, o.SecondaryOwners
FROM VehicleRegistration AS r, @r.Owners AS o
WHERE r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

SELECT — Innerer Zusammenschluss

```
SELECT v.Make, v.Model, r.Owners
FROM VehicleRegistration AS r INNER JOIN Vehicle AS v
ON r.VIN = v.VIN
WHERE r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

SELECT — Ermittelt die Dokument-ID mithilfe der BY-Klausel

```
SELECT r_id FROM VehicleRegistration AS r BY r_id
WHERE r.VIN = '1HVBBAANXWH544237'
```

Systemdefinierte Ansichten

QLDB unterstützt die folgenden systemdefinierten Ansichten einer Tabelle.

Anzeigen	Beschreibung
table_name	Die Standardbenutzeransicht einer Tabelle, die nur den aktuellen Status Ihrer Benutzerdaten enthält.
_ql_committed_ *table_name*	Die vollständige systemdefinierte, festgeschriebene Ansicht einer Tabelle, die den aktuellen Status sowohl Ihrer Benutzerdaten als auch der vom System generierten Metadaten, z. B. einer Dokument-ID, enthält.

Anzeigen	Beschreibung
`history(table_name )`	Die integrierte Verlaufsfunktion, die den vollständigen Revisionsverlauf einer Tabelle zurückgibt.

Grundlegende Syntaxregeln

QLDB unterstützt die folgenden grundlegenden Syntaxregeln für PartiQL.

Zeichen	Beschreibung
'	Einfache Anführungszeichen bezeichnen Zeichenkettenwerte oder Feldnamen in Amazon Ion-Strukturen.
"	Doppelte Anführungszeichen stehen für Bezeichner in Anführungszeichen, wie z. B. ein reserviertes Wort, das als Tabellenname verwendet wird.
`	Backticks stehen für Ion-Literalwerte.
.	Die Punktnotation greift auf Feldnamen einer übergeordneten Struktur zu.
[ ]	Eckige Klammern definieren ein Ion `list` oder bezeichnen eine auf Null basierende Ordinalzahl für eine bestehende Liste.
{ }	Geschweifte Klammern definieren ein Ion. `struct`
<< >>	Doppelte spitze Klammern definieren eine PartiQL-Tasche, bei der es sich um eine ungeordnete Sammlung handelt. Sie verwenden eine Tasche, um mehrere Dokumente in eine Tabelle einzufügen.
Groß-/Kleinschreibung	Bei allen QLDB-Systemobjektnamen — einschließlich Feld- und Tabellennamen — wird zwischen Groß- und Kleinschreibung unterschieden.

Tastenkombinationen für den PartiQL-Editor

Der PartiQL-Editor auf der QLDB-Konsole unterstützt die folgenden Tastenkombinationen.

Action	macOS	Windows
Ausführen	Cmd+Return	Ctrl+Enter
Kommentar	Cmd+/	Ctrl+/
Löschen Sie	Cmd+Shift+Delete	Ctrl+Shift+Delete

Zugreifen auf Amazon QLDB mithilfe der AWS CLI (nur Verwaltungs-API)

Sie können die AWS Command Line Interface (AWS CLI) verwenden, um mehrere AWS-Services von der Befehlszeile aus zu steuern und sie mithilfe von Skripten zu automatisieren. Sie können das AWS CLI bei Bedarf für einmalige Operationen verwenden. Sie können es auch verwenden, um Amazon QLDB-Operationen in Utility-Skripte einzubetten.

Für CLI-Zugriff benötigen Sie eine Zugriffsschlüssel-ID und einen geheimen Zugriffsschlüssel. Verwenden Sie möglichst temporäre Anmeldeinformationen anstelle langfristiger Zugriffsschlüssel. Temporäre Anmeldeinformationen bestehen aus einer Zugriffsschlüssel-ID, einem geheimen Zugriffsschlüssel und einem Sicherheits-Token, das angibt, wann die Anmeldeinformationen ablaufen. Weitere Informationen finden Sie unter Verwenden von temporären Anmeldeinformationen mit AWS-Ressourcen im IAM-Benutzerhandbuch.

Eine vollständige Liste und Anwendungsbeispiele aller für QLDB verfügbaren Befehle in der finden Sie in der AWS CLI AWS CLIBefehlsreferenz.

> **Note**
>
> Die unterstützt AWS CLI nur die `qldb` Verwaltungs-API-Operationen, die in der aufgeführt sind. Amazon QLDB API-Referenz Diese API wird nur für die Verwaltung von Ledger-Ressourcen und für nicht transaktionale Datenoperationen verwendet.
> Informationen zum Ausführen von Datentransaktionen mit der `qldb-session` API über eine Befehlszeilenschnittstelle finden Sie unter. Zugriff auf Amazon QLDB mithilfe der QLDB-Shell (nur Daten-API)

Themen

- Installation und Konfiguration des AWS CLI
- Verwenden von AWS CLI mit QLDB

Installation und Konfiguration des AWS CLI

Das AWS CLI läuft unter Linux, MacOS oder Windows. Informationen zur Installation und Konfiguration finden Sie in den folgenden Anweisungen im AWS Command Line InterfaceBenutzerhandbuch:

1. Installation oder Aktualisierung der neuesten Version der AWS CLI
2. Schnelleinrichtung

Verwenden von AWS CLI mit QLDB

Das Befehlszeilenformat besteht aus einem Amazon QLDB-Operationsnamen, gefolgt von den Parametern für diesen Vorgang. Die AWS CLI unterstützt eine Syntax-Kurznotation für die Parameterwerte, zusätzlich zu JSON.

Verwenden Sie`help`, um alle verfügbaren Befehle in QLDB aufzulisten:

```
aws qldb help
```

Sie können auch `help` verwenden, um einen bestimmten Befehl zu beschreiben und mehr über seine Nutzung zu erfahren:

```
aws qldb create-ledger help
```

Um beispielsweise eine Ledger zu erstellen:

```
aws qldb create-ledger --name my-example-ledger --permissions-mode STANDARD
```

Zugriff auf Amazon QLDB mithilfe der QLDB-Shell (nur Daten-API)

Amazon QLDB bietet eine Befehlszeilen-Shell für die Interaktion mit der Transaktionsdaten-API. Mit der QLDB-Shell können Sie PartiQL-Anweisungen für Ledger-Daten ausführen.

Die neueste Version dieser Shell wurde in Rust geschrieben und ist Open Source im GitHub Repository awslabs/amazon-qldb-shell im `main` Standard-Branch. Die Python-Version (v1) ist ebenfalls weiterhin für die Verwendung im selben Repository auf dem `master` Zweig verfügbar.

> **Note**
>
> Die Amazon QLDB-Shell unterstützt nur die `qldb-session` Transaktionsdaten-API. Diese API wird nur für die Ausführung von PartiQL-Anweisungen in einem QLDB-Ledger verwendet. Informationen zur Interaktion mit den `qldb` Management-API-Vorgängen mithilfe einer Befehlszeilenschnittstelle finden Sie unterZugreifen auf Amazon QLDB mithilfe der AWS CLI (nur Verwaltungs-API).

Dieses Tool ist nicht dafür vorgesehen, in eine Anwendung integriert oder für Produktionszwecke verwendet zu werden. Das Ziel dieses Tools ist es, Sie schnell mit QLDB und PartiQL experimentieren zu lassen.

In den folgenden Abschnitten werden die ersten Schritte mit der QLDB-Shell beschrieben.

Themen

Voraussetzungen

Bevor Sie mit der QLDB-Shell beginnen, müssen Sie Folgendes tun:

1. Befolgen Sie die AWS-Einrichtungsanweisungen unter Zugreifen auf Amazon QLDB. Diese umfasst die folgenden Funktionen:

   1. Registrieren Sie sich fürAWS.
   2. Erstellen Sie einen Benutzer mit den entsprechenden QLDB-Berechtigungen.
   3. Erteilen programmgesteuerten Zugriffs
2. Richten Sie IhreAWS Anmeldeinformationen und Ihre Standardeinstellungen einAWS-Region. Anweisungen finden Sie unter Konfigurationsgrundlagen im AWS Command Line InterfaceBenutzerhandbuch.

   Eine vollständige Liste der verfügbaren Regionen finden Sie unter Amazon QLDB-Endpunkte und Kontingente in der Allgemeine AWS-Referenz.
3. Erstellen Sie für alle Ledger im`STANDARD` Berechtigungsmodus IAM-Richtlinien, die Ihnen die Berechtigung gewähren, PartiQL-Anweisungen in den entsprechenden Tabellen auszuführen. Informationen zum Erstellen dieser Richtlinien finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

Installation der Shell

Informationen zur Installation der neuesten Version der QLDB-Shell finden Sie in der README.md-Datei unter GitHub. QLDB stellt vorgefertigte Binärdateien für Linux, macOS und Windows im Bereich Releases des GitHub Repositorys bereit.

Für macOS ist die Shell in den`aws/tap` Homebrew-Tap integriert. Führen Sie die folgenden Befehle aus, um die Shell unter macOS mit Homebrew zu installieren.

```
$ xcode-select --install # Required to use Homebrew
$ brew tap aws/tap # Add AWS as a Homebrew tap
$ brew install qldbshell
```

Konfiguration

Nach der Installation lädt die Shell die Standardkonfigurationsdatei, die sich`$XDG_CONFIG_HOME/qldbshell/config.ion` während der Initialisierung unter befindet. Unter Linux und macOS befindet sich diese Datei normalerweise unter`~/.config/qldbshell/config.ion`. Wenn eine solche Datei nicht existiert, wird die Shell mit Standardeinstellungen ausgeführt.

Sie können eine`config.ion` Datei nach der Installation manuell erstellen. Diese Konfigurationsdatei verwendet das Amazon Ion-Datenformat. Im Folgenden wird ein Beispiel für eine`config.ion` Minimaldatei gezeigt.

```
{
  default_ledger: "my-example-ledger"
}
```

Wenn er in Ihrer Konfigurationsdatei`default_ledger` nicht gesetzt ist, ist der`--ledger` Parameter erforderlich, wenn Sie die Shell aufrufen. Eine vollständige Liste der Konfigurationsoptionen finden Sie in der README.md-Datei unter GitHub.

Aufrufen der Shell

Um die QLDB-Shell auf Ihrem Befehlszeilenterminal für ein bestimmtes Ledger aufzurufen, führen Sie den folgenden Befehl aus. *`my-example-ledger`*Ersetzen Sie es durch Ihren Hauptbuchnamen.

```
$ qldb --ledger my-example-ledger
```

Dieser Befehl stellt eine Verbindung zu Ihrer Standardeinstellung herAWS-Region. Um die Region explizit anzugeben, können Sie den Befehl mit dem`--qldb-session-endpoint` Parameter`--region` or ausführen, wie im folgenden Abschnitt beschrieben.

Nach dem Aufrufen einer`qldb` Shell-Sitzung können Sie die folgenden Eingabetypen eingeben:

- Shell-Befehle
- Einzelne PartiQL-Anweisungen in separaten Transaktionen
- Mehrere PartiQL-Anweisungen innerhalb einer Transaktion

Shell-Parameter

Um eine vollständige Liste der verfügbaren Flags und Optionen zum Aufrufen einer Shell zu erhalten, führen Sie den`qldb` Befehl mit dem`--help` Flag wie folgt aus.

```
$ qldb --help
```

Im Folgenden sind einige wichtige Flags und Optionen für den`qldb` Befehl aufgeführt. Sie können diese optionalen Parameter hinzufügen, um das AnmeldeinformationsprofilAWS-Region, den Endpunkt, das Ergebnisformat und andere Konfigurationsoptionen zu überschreiben.

Usage

```
$ qldb [FLAGS] [OPTIONS]
```

FLAGGEN

`-h`, **`--help`**

Druckt Hilfeinformationen aus.

`-v`, **`--verbose`**

Konfiguriert die Ausführlichkeit der Protokollierung. Standardmäßig protokolliert die Shell nur Fehler. Um den Ausführlichkeitsgrad zu erhöhen, wiederholen Sie dieses Argument (z. B.`-vv`). Die höchste Stufe`-vvv` entspricht der`trace` Ausführlichkeit.

`-V`, **`--version`**

Druckt Versionsinformationen aus.

OPTIONS

`-l--ledger`, ***`BUCHNAME`***

Der Name des Ledgers, mit dem eine Verbindung hergestellt werden soll. Dies ist ein erforderlicher Shell-Parameter, falls er in Ihrer`config.ion` Datei`default_ledger` nicht gesetzt ist. In dieser Datei können Sie zusätzliche Optionen festlegen, z. B. die Region.

`-c`,**`--config`** ***`KONFIGURATIONSDATEI`***

Die Datei, in der Sie alle Shell-Konfigurationsoptionen definieren können. Formatierungsdetails und eine vollständige Liste der Konfigurationsoptionen finden Sie in der Datei README.md unter GitHub.

`-f`, **`--format ion|table`**

Das Ausgabeformat Ihrer Abfrageergebnisse. Der Standardwert ist `ion`.

`-p`,**`--profile`** ***`PROFIL`***

Der Speicherort IhresAWS Anmeldeinformationsprofils, das für die Authentifizierung verwendet werden soll.

Wenn es nicht angegeben wird, verwendet die Shell das AWS-Standardprofil in `~/.aws/credentials`.

`-r`,`--region` *`REGIONALCODE`*

DerAWS-Region Code des QLDB-Ledgers, zu dem eine Verbindung hergestellt werden soll. Zum Beispiel: `us-east-1`.

Wenn nicht angegeben, stellt die Shell eine Verbindung zu Ihrer Standardeinstellung her,AWS-Region wie in IhremAWS Profil angegeben.

`-s`,`--qldb-session-endpoint` *`QLDB_SITZUNGSENDPUNKT`*

Der`qldb-session` API-Endpunkt, zu dem eine Verbindung hergestellt werden soll.

Eine vollständige Liste der verfügbaren QLDB-Regionen und Endpunkte finden Sie unter Amazon QLDB-Endpunkte und -Kontingente in der Allgemeine AWS-Referenz.

Befehlsreferenz

Nachdem Sie ein`qldb` Sitzung aufgerufen haben, unterstützt die Shell die folgenden Schlüssel und Datenbankbefehle:

Shell-Schlüssel

Schlüssel	Beschreibung der Funktion
Enter	Führt die Anweisung aus.
Escape+Enter (macOS, Linux) Shift+Enter (Windows)	Beginnt eine neue Zeile, um eine Anweisung einzugeben, die sich über mehrere Zeilen erstreckt. Sie können den Eingabetext auch mit mehreren Zeilen kopieren und in die Shell einfügen. Anweisungen zur EinrichtungOption stattEscape als Metaschlüssel in macOS finden Sie auf der OS X Daily-Website.
Ctrl+C	Bricht den aktuellen Befehl ab.
Ctrl+D	Signalisiert das Ende der Datei (EOF) und beendet die aktuelle Ebene der Shell. Beendet die Shell, falls es sich nicht um eine aktive

Schlüssel	Beschreibung der Funktion
	Transaktion handelt. Bricht in einer aktiven Transaktion die Transaktion ab.

Shell-Datenbankbefehle

Befehl	Beschreibung der Funktion
`help`	Zeigt die Hilfeinformationen an.
`begin`	Startet eine Transaktion.
`start transaction`	
`commit`	Überträgt Ihre Transaktion in das Journal des Ledgers.
`abort`	Stoppt Ihre Transaktion und lehnt alle von Ihnen vorgenommenen Änderungen ab.
`exit`	Verlässt die Shell.
`quit`	

> **Note**
>
> Bei allen QLDB-Shell-Befehlen wird die Groß- und Kleinschreibung nicht berücksichtigt.

Einzelne Kontoauszüge ausführen

Mit Ausnahme der in README.md aufgeführten Datenbankbefehle und Shell-Metabefehle interpretiert die Shell jeden Befehl, den Sie eingeben, als separate PartiQL-Anweisung. Standardmäßig aktiviert die Shell den`auto-commit` Modus. Dieser Modus ist konfigurierbar.

Im`auto-commit` Modus führt die Shell implizit jede Anweisung in ihrer eigenen Transaktion aus und überträgt die Transaktion automatisch, wenn keine Fehler gefunden werden. Das bedeutet, dass

Sie nicht jedes Mal, wenn Sie eine Anweisung ausführen,`commit` manuell`start transaction` (oder`begin`) ausführen müssen.

Verwalten von Transaktionen

Alternativ können Sie mit der QLDB-Shell Transaktionen manuell steuern. Sie können mehrere Anweisungen innerhalb einer Transaktion interaktiv oder nicht interaktiv ausführen, indem Sie Befehle und Anweisungen nacheinander stapeln.

Interaktive Transaktionen

Führen Sie die folgenden Schritte aus, um eine interaktive Transaktion auszuführen.

1. Um eine Transaktion zu starten, geben Sie den`begin` Befehl ein.

   ```
   qldb> begin
   ```

   Nachdem Sie eine Transaktion gestartet haben, zeigt die Shell die folgende Befehlszeile an.

   ```
   qldb *>
   ```

2. Dann wird jede Anweisung, die Sie eingeben, in derselben Transaktion ausgeführt.
   - Zum Beispiel können Sie eine einzelne Anweisung wie folgt ausführen.

     ```
     qldb *> SELECT * FROM Vehicle WHERE VIN = '1N4AL11D75C109151'
     ```

     Nachdem Sie gedrückt habenEnter, zeigt die Shell die Ergebnisse der Anweisung an.
   - Sie können auch mehrere Anweisungen oder Befehle eingeben, die durch ein Semikolon (`;`) als Trennzeichen getrennt sind, wie folgt.

     ```
     qldb *> SELECT * FROM Vehicle WHERE VIN = '1N4AL11D75C109151'; commit
     ```

3. Geben Sie einen der folgenden Befehle ein, um die Transaktion zu beenden.
   - Geben Sie den`commit` Befehl ein, um Ihre Transaktion im Journal des Ledgers zu speichern.

     ```
     qldb *> commit
     ```

- Geben Sie den`abort` Befehl ein, um Ihre Transaktion zu beenden und alle von Ihnen vorgenommenen Änderungen abzulehnen.

```
qldb *> abort
transaction was aborted
```

Timeout-Limit für Transaktionen

Eine interaktive Transaktion hält sich an das Transaktions-Timeout-Limit von QLDB. Wenn Sie eine Transaktion nicht innerhalb von 30 Sekunden nach dem Start bestätigen, läuft QLDB die Transaktion automatisch ab und lehnt alle während der Transaktion vorgenommenen Änderungen ab.

Anstatt die Anweisungsergebnisse anzuzeigen, zeigt die Shell dann eine Ablauffehlermeldung an und kehrt zur normalen Befehlszeile zurück. Um es erneut zu versuchen, müssen Sie den`begin` Befehl erneut eingeben, um eine neue Transaktion zu starten.

```
transaction failed after 1 attempts, last error: communication failure:
 Transaction 2UMpiJ5hh7WLjVgEiMLOoO has expired
```

Nicht interaktive Transaktionen

Sie können eine vollständige Transaktion mit mehreren Anweisungen ausführen, indem Sie Befehle und Anweisungen wie folgt nacheinander stapeln.

```
qldb> begin; SELECT * FROM Vehicle WHERE VIN = '1N4AL11D75C109151'; SELECT * FROM
 Person p, DriversLicense l WHERE p.GovId = l.LicenseNumber; commit
```

Sie müssen jeden Befehl und jede Anweisung durch ein Semikolon (`;`) als Trennzeichen trennen. Wenn eine Anweisung in der Transaktion nicht gültig ist, lehnt die Shell die Transaktion automatisch ab. Die Shell fährt mit den nachfolgenden Anweisungen, die Sie eingegeben haben, nicht fort.

Sie können auch mehrere Transaktionen einrichten.

```
qldb> begin; statement1; commit; begin; statement2; statement3; commit
```

Ähnlich wie im vorherigen Beispiel fährt die Shell nicht mit den nachfolgenden Transaktionen oder Kontoauszügen fort, die Sie eingegeben haben, wenn eine Transaktion fehlschlägt.

Wenn Sie eine Transaktion nicht beenden, wechselt die Shell in den interaktiven Modus und fordert Sie auf, den nächsten Befehl oder die nächste Anweisung einzugeben.

```
qldb> begin; statement1; commit; begin
qldb *>
```

Verlassen der Shell

Um die aktuelle`qldb` Shell-Sitzung zu beenden, geben Sie den`quit` Befehl`exit` or ein, oder verwenden Sie die TastenkombinationCtrl +,D wenn sich die Shell nicht in einer Transaktion befindet.

```
qldb> exit
$
```

```
qldb> quit
$
```

Beispiel

Informationen zum Schreiben von PartiQL-Anweisungen in QLDB finden Sie unterAmazon QLDB PartiQL-Referenz.

Example

Das folgende Beispiel zeigt eine typische Folge grundlegender Befehle.

> **Note**
>
> Die QLDB-Shell führt jede PartiQL-Anweisung in diesem Beispiel in einer eigenen Transaktion aus.
> In diesem Beispiel wird unterstellt, dass der Ledger `test-ledger` bereits vorhanden und aktiv ist.

```
$ qldb --ledger test-ledger --region us-east-1

qldb> CREATE TABLE TestTable
qldb> INSERT INTO TestTable `{"Name": "John Doe"}`
```

```
qldb> SELECT * FROM TestTable
qldb> DROP TABLE TestTable
qldb> exit
```

Zugreifen auf Amazon QLDB mithilfe der API

Sie können das AWS Management Console und das AWS Command Line Interface (AWS CLI) verwenden, um interaktiv mit Amazon QLDB zu arbeiten. Um jedoch das Beste aus QLDB herauszuholen, können Sie Anwendungscode mit einem QLDB-Treiber oder einem AWS SDK schreiben, um mithilfe der APIs mit Ihrem Ledger zu interagieren.

Der Treiber ermöglicht es Ihrer Anwendung, mithilfe der Transaktionsdaten-API mit QLDB zu interagieren. Das AWS SDK unterstützt die Interaktion mit der QLDB-Ressourcenmanagement-API. Weitere Informationen zu diesen APIs finden Sie unter. Amazon QLDB API-Referenz

Der Treiber bietet Unterstützung für QLDB in Java, .NET, Go, Node.js und Python. Informationen für einen schnellen Einstieg in diese Sprachen finden Sie unter Erste Schritte mit dem Amazon-QLDB-Treiber.

Bevor Sie einen QLDB-Treiber oder ein AWS SDK in Ihrer Anwendung verwenden können, müssen Sie programmatischen Zugriff gewähren. Weitere Informationen finden Sie unter Erteilen programmgesteuerten Zugriffs.

Erste Schritte mit der Amazon QLDB-Konsole

Dieses Tutorial führt Sie durch die Schritte, um Ihr erstes Amazon QLDB-Ledger zu erstellen und es mit Tabellen und Beispieldaten zu füllen. Der Beispiel-Ledger, den Sie in diesem Szenario anlegen, ist eine Datenbank für eine Anwendung der Straßenverkehrsbehörde, in der die vollständigen Verlaufsinformationen über Fahrzeugzulassungen nachverfolgt werden.

Die Historie einer Anlage ist ein häufiger Anwendungsfall für QLDB, da sie eine Vielzahl von Szenarien und Vorgängen beinhaltet, die die Nützlichkeit einer Ledger-Datenbank unterstreichen. Mit QLDB können Sie in einer dokumentorientierten Datenbank, die SQL-ähnliche Abfragefunktionen unterstützt, direkt auf den vollständigen Verlauf der Änderungen an Ihren Daten zugreifen, ihn abfragen und überprüfen.

Während Sie dieses Tutorial durcharbeiten, wird in den folgenden Themen erläutert, wie Sie Fahrzeugregistrierungen hinzufügen, ändern und den Verlauf der Änderungen an diesen Zulassungen einsehen können. In diesem Handbuch erfahren Sie auch, wie Sie ein Registrierungsdokument kryptografisch verifizieren können. Abschließend werden Ressourcen bereinigt und das Musterbuch gelöscht.

Jeder Schritt in der Anleitung enthält Anweisungen für die Verwendung der AWS Management Console.

Themen

Voraussetzungen und Überlegungen zum Tutorial

Bevor Sie mit diesem Amazon QLDB-Tutorial beginnen, müssen Sie sicherstellen, dass Sie die folgenden Voraussetzungen erfüllen:

1. Folgen Sie denAWS Einrichtungsanweisungen unterZugreifen auf Amazon QLDB, falls noch nicht geschehen. Zu diesen Schritten gehören die RegistrierungAWS und das Erstellen eines Administratorbenutzers.
2. Folgen SieEinrichten von Berechtigungen den Anweisungen unter Einrichten von IAM-Berechtigungen für Ihre QLDB-Ressourcen. Um alle Schritte dieses Tutorials auszuführen, benötigen Sie vollständigen administrativen Zugriff auf Ihre Ledger-Ressourcen über dieAWS Management Console.

   > **Note**
   >
   > Wenn Sie bereits als Benutzer mit vollständigenAWS Administratorberechtigungen angemeldet sind, können Sie diesen Schritt überspringen.

3. (Optional) QLDB verschlüsselt ruhende Daten mit einem Schlüssel inAWS Key Management Service (AWS KMS). Sie können eine der folgenden Optionen auswählenAWS KMS keys:
   - AWSeigener KMS-Schlüssel — Einen KMS-Schlüssel verwenden, der gehört und inAWS Ihrem Namen verwaltet wird. Dies ist die Standardoption und erfordert keine zusätzliche Einrichtung.
   - Kundenverwalteter KMS-Verschlüsselungsschlüssel — Einen symmetrischen KMS-Verschlüsselungsschlüssel in Ihrem Konto verwenden, den Sie erstellen, besitzen und verwalten. QLDB unterstützt keine asymmetrischen Schlüssel.

     Für diese Option müssen Sie einen KMS-Schlüssel erstellen oder einen vorhandenen Schlüssel in Ihrem Konto verwenden. Anweisungen zum Erstellen eines vom Kunden verwalteten Schlüssels finden Sie im AWS Key Management ServiceEntwicklerhandbuch unter Erstellen von KMS-Schlüsseln mit symmetrischer Verschlüsselung.

     Sie können einen vom Kunden verwalteten KMS-Schlüssel verwalteter KMS-Schlüssel verwalteter KMS-Schlüssel (ARN) aus. Weitere Informationen finden Sie unter Schlüsselkennungen (KeyId) im AWS Key Management ServiceEntwicklerhandbuch.

> **Note**
>
> Regionsübergreifende Schlüssel werden nicht unterstützt. Der angegebene KMS-Schlüssel muss sich im gleichenAWS-Region wie in Ihrem Ledger befinden.

Einrichten von Berechtigungen

In diesem Schritt richten Sie über die Konsole vollständige Zugriffsberechtigungen für alle QLDB-Ressourcen in Ihrem einAWS-Konto. Verwenden Sie dieAWS verwaltete Richtlinie AmazonQLDB, um diese Berechtigungen schnell zu gewährenConsoleFullAccess.

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center-Benutzerhandbuch.
- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

> **Important**
>
> Für die Zwecke dieses Tutorials gewähren Sie sich vollen administrativen Zugriff auf alle QLDB-Ressourcen. Für produktive Anwendungsfälle sollten Sie sich jedoch an die bewährte Sicherheitsmethode halten, d. h. es werden die geringsten Rechte erteilt, d. h. es werden nur die Berechtigungen erteilt, die für die Durchführung einer Aufgabe erforderlich sind. Beispiele finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

Um ein Ledger namens zu erstellen`vehicle-registration`, fahren Sie mit fortSchritt 1: Erstellen eines neuen Ledgers.

Schritt 1: Erstellen eines neuen Ledgers

In diesem Schritt erstellen Sie ein neues Amazon QLDB-Ledger namens`vehicle-registration`. Anschließend bestätigen Sie, dass der Status des Ledgers Active (Aktiv)ist. Sie können auch alle Tags überprüfen, die Sie dem Ledger hinzugefügt haben.

Wenn Sie einen Ledger erstellen, ist der Löschschutz standardmäßig aktiviert. Der Löschschutz ist eine Funktion in QLDB, die verhindert, dass Ledgers von einem beliebigen Benutzer gelöscht werden. Sie können den Löschschutz deaktivieren, wenn Sie ein Ledger mithilfe der QLDB-API oder derAWS Command Line Interface (AWS CLI) erstellen.

So erstellen Sie einen neuen Ledger

1. Melden Sie sich bei derAWS Management Console und öffnen Sie die Amazon QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich Getting started (Erste Schritte) aus.
3. Wählen Sie auf der Karte Create your first Ledger (Ihren ersten Ledger erstellen) die Option Create Ledger (Ledger erstellen) aus.
4. Gehen Sie auf der Seite Create Ledger (Ledger erstellen) wie folgt vor:

   - Buchinformationen — Der Name des Buchs sollte vorab mit ausgefüllt werden**`vehicle-registration`**.
   - Berechtigungsmodus — Der Berechtigungsmodus, der dem Ledger zugewiesen werden soll. Wählen Sie eine der folgenden Optionen:
     - Allow all — Ein Legacy-Berechtigungsmodus, der die Zugriffskontrolle mit Granularität auf API-Ebene für Ledgers ermöglicht.

       Dieser Modus ermöglicht Benutzern, die `SendCommand`-API-Berechtigung für dieses Ledger zum Ausführen aller PartiQL-Befehle (daher `ALLOW_ALL`) auf beliebigen Tabellen im angegebenen Ledger auszuführen. Dieser Modus ignoriert alle IAM-Berechtigungs-Richtlinien auf Tabellen- oder Befehls-Ebene, die Sie für das Ledger erstellen.
     - Standard — (Empfohlen) Ein Berechtigungsmodus, der Zugriffskontrolle mit feinerer Granularität für Ledgers, Tabellen und PartiQL-Befehle ermöglicht. Wir empfehlen dringend,

diesen Berechtigungsmodus zu verwenden, um die Sicherheit Ihrer Ledger-Daten zu maximieren.

Standardmäßig verweigert dieser Modus alle Anforderungen zur Ausführung von PartiQL-Befehlen für Tabellen in diesem Ledger. Um PartiQL-Befehle zuzulassen, müssen Sie IAM-Berechtigungsrichtlinien für bestimmte Tabellenressourcen und PartiQL-Aktionen erstellen, zusätzlich zu der`SendCommand` API-Berechtigung für das Ledger. Weitere Informationen finden Sie unter Erste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

- Verschlüsseln von Daten im RuhezustandAWS Key Management Service (AWS KMS), der für die Verschlüsselung von Daten im Ruhezustand () verwendet werden soll. Wählen Sie eine der folgenden Optionen:
  - EinenAWS eigenen KMS-Schlüssel verwenden — Einen KMS-Schlüssel verwenden, der gehört und inAWS Ihrem Namen verwaltet wird. Dies ist die Standardoption und erfordert keine zusätzliche Einrichtung.
  - Einen anderenAWS KMS Schlüssel auswählen — Einen symmetrischen KMS-Verschlüsselungsschlüssel in Ihrem Konto verwenden, den Sie erstellen, besitzen und verwalten.

    Um einen neuen Schlüssel mithilfe derAWS KMS Konsole zu erstellen, wählen Sie AWS KMSSchlüssel erstellen. Weitere Informationen finden Sie unter Erstellen symmetrischer KMS-Verschlüsselungsschlüssel im AWS Key Management Service-Entwicklerhandbuch.

    Um einen vorhandenen KMS-Schlüssel zu verwenden, wählen Sie einen aus der Dropdownliste aus oder geben Sie einen KMS-Schlüssel-ARN an.
- Tags — (Optional) Fügen Sie dem Ledger Metadaten hinzu, indem Sie Tags als Schlüssel-Wert-Paare anfügen. Sie können Ihrem Ledger Tags hinzufügen, um sie zu organisieren und zu identifizieren. Weitere Informationen finden Sie unter Markieren von Amazon-QLDB Ressourcen.

  Wählen Sie Add tag (Tag hinzufügen) aus und geben Sie dann beliebige Schlüssel-Wert-Paare ein.

5. Wenn Sie die gewünschten Einstellungen vorgenommen haben, wählen Sie Create ledger (Ledger erstellen) aus.

> **Note**
>
> Sie können auf Ihr QLDB-Ledger zugreifen, wenn sein Status Aktiv wird. Dies kann mehrere Minuten dauern.

6. Suchen Sie in der Liste der Ledgers (Ledger) `vehicle-registration` und vergewissern Sie sich, dass der Status des Ledgers Active (Aktiv) ist.
7. (Optional) Wählen Sie den Namen des Ledgers `vehicle-registration` aus. Bestätigen Sie auf der Seite mit den Details des vehicle-registration-Ledgers, dass alle Tags, die Sie dem Ledger hinzugefügt haben, auf der Karte Tags angezeigt werden. Sie können Ihre Ledger-Tags auch über diese Konsolenseite bearbeiten.

Fahren Sie zum Erstellen von Tabellen im `vehicle-registration`-Ledger mit Schritt 2: Erstellen Sie Tabellen, Indizes und Beispieldaten in einem Ledger fort.

Schritt 2: Erstellen Sie Tabellen, Indizes und Beispieldaten in einem Ledger

Wenn Ihr Amazon QLDB-Ledger aktiv ist, können Sie damit beginnen, Tabellen für Daten über Fahrzeuge, deren Besitzer und deren Zulassungsinformationen zu erstellen. Nach dem Erstellen der Tabellen und Indizes können Sie Daten in diese laden.

In diesem Schritt erstellen Sie vier Tabellen im `vehicle-registration`-Ledger:

- `VehicleRegistration`
- `Vehicle`
- `Person`
- `DriversLicense`

Außerdem erzeugen Sie die folgenden Indizes.

Tabellenname	Feld
`VehicleRegistration`	`VIN`

Tabellenname	Feld
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicensePlateNumber`
`DriversLicense`	`PersonId`

Sie können die QLDB-Konsole verwenden, um diese Tabellen automatisch mit Indizes zu erstellen und sie mit Beispieldaten zu laden. Oder Sie können den PartiQL-Editor auf der Konsole verwenden, um jede PartiQL-Anweisung manuell auszuführen step-by-step.

Automatische Option

So erstellen Sie Tabellen, Indizes und Beispieldaten

1. Öffnen Sie die Amazon QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich Getting started (Erste Schritte) aus.
3. Wählen Sie unter Automatic (Automatisch) auf der Karte Sample application data (Beispielanwendungsdaten) `vehicle-registration` in der Liste der Ledger aus.
4. Wählen Sie Load sample data (Beispieldaten laden) aus.

   Wenn der Vorgang erfolgreich abgeschlossen wird, zeigt die Konsole die Nachricht Sample data loaded (Beispieldaten geladen) an.

   Dieses Skript führt alle Anweisungen in einer einzigen Transaktion aus. Falls ein Teil der Transaktion fehlschlägt, wird für jede Anweisung ein Rollback ausgeführt und eine entsprechende Fehlermeldung wird angezeigt. Sie können den Vorgang nach der Behebung etwaiger Probleme erneut versuchen.

   > **Note**
   >
   > - Ein möglicher Grund für das Fehlschlagen der Transaktion ist der Versuch, doppelte Tabellen zu erstellen. Ihre Anfrage zum Laden von Beispieldaten schlägt fehl,

wenn einer der folgenden Tabellennamen in Ihrem Ledger bereits vorhanden ist: `VehicleRegistration`, `Vehicle`, `Person` und `DriversLicense`.

Versuchen Sie stattdessen, diese Beispieldaten in einen leeren Ledger zu laden.

- Dieses Skript führt parametrisierte `INSERT`-Anweisungen aus. Diese PartiQL-Anweisungen werden also in Ihren Journal-Blöcken mit Bindungsparametern anstelle der Literaldaten aufgezeichnet. Sie können beispielsweise die folgende Anweisung in einem Journal-Block anzeigen, wobei das Fragezeichen (?) ein Variablenplatzhalter für den Dokumentinhalt ist.

```
INSERT INTO Vehicle ?
```

Manuelle Option

Dazu fügen Sie Dokumente in `VehicleRegistration` mit einem leeren `PrimaryOwner`-Feld und in `DriversLicense` mit einem leeren `PersonId`-Feld ein. Später füllen Sie diese Felder mit dem vom System zugewiesenen Dokument `id` aus der `Person`-Tabelle.

> **Tip**
>
> Als bewährte Methode verwenden Sie dieses `id`-Dokumentmetadatenfeld als Fremdschlüssel. Weitere Informationen finden Sie unter Metadaten von Dokumenten abfragen.

So erstellen Sie Tabellen, Indizes und Beispieldaten

1. Öffnen Sie die Amazon QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich den PartiQL-Editor aus.
3. Wählen Sie den `vehicle-registration`-Ledger aus.
4. Erstellen Sie zunächst vier Tabellen. QLDB unterstützt offene Inhalte und erzwingt kein Schema, sodass Sie keine Attribute oder Datentypen angeben.

   Geben Sie im Fenster des Abfrageeditors die folgende Anweisung ein und klicken Sie dann auf Run (Ausführen). Um die Anweisung auszuführen, können Sie auch die Tastenkombination

Ctrl+Enter für Windows oder Cmd+Return für macOS verwenden. Weitere Tastenkombinationen finden Sie unter Tastenkombinationen für den PartiQL-Editor.

```
CREATE TABLE VehicleRegistration
```

Wiederholen Sie diesen Schritt für jede der folgenden Aussagen.

```
CREATE TABLE Vehicle
```

```
CREATE TABLE Person
```

```
CREATE TABLE DriversLicense
```

5. Erstellen Sie als Nächstes Indizes, die die Abfrageleistung für jede Tabelle optimieren.

> **Important**
>
> QLDB benötigt einen Index, um ein Dokument effizient nachschlagen zu können. Ohne Index muss QLDB beim Lesen von Dokumenten einen vollständigen Tabellenscan durchführen. Dies kann bei großen Tabellen zu Leistungsproblemen führen, einschließlich Parallelitätskonflikten und Transaktions-Timeouts.
> Um Tabellenscans zu vermeiden, müssen Sie Anweisungen mit einer`WHERE` Prädikatklausel unter Verwendung eines Gleichheitsoperators (`=`oder`IN`) für ein indiziertes Feld oder eine Dokument-ID ausführen. Weitere Informationen finden Sie unter Optimieren der Abfrageleistung.

Geben Sie im Fenster des Abfrageeditors die folgende Anweisung ein und klicken Sie dann auf Run (Ausführen).

```
CREATE INDEX ON VehicleRegistration (VIN)
```

Wiederholen Sie diesen Schritt für die folgenden Aussagen.

```
CREATE INDEX ON VehicleRegistration (LicensePlateNumber)
```

```
CREATE INDEX ON Vehicle (VIN)
```

```
CREATE INDEX ON Person (GovId)
```

```
CREATE INDEX ON DriversLicense (LicensePlateNumber)
```

```
CREATE INDEX ON DriversLicense (PersonId)
```

6. Nach dem Erstellen Ihrer Indizes können Sie mit dem Laden von Daten in Ihre Tabellen beginnen. Mit diesem Schritt fügen Sie Dokumente in die `Person`-Tabelle mit personenbezogenen Angaben zu Fahrzeughaltern ein, die der Ledger verfolgt.

   Geben Sie im Fenster des Abfrageeditors die folgende Anweisung ein und klicken Sie dann auf Run (Ausführen).

```
INSERT INTO Person
<< {
    'FirstName' : 'Raul',
    'LastName' : 'Lewis',
    'DOB' : `1963-08-19T`,
    'GovId' : 'LEWISR261LL',
    'GovIdType' : 'Driver License',
    'Address' : '1719 University Street, Seattle, WA, 98109'
},
{
    'FirstName' : 'Brent',
    'LastName' : 'Logan',
    'DOB' : `1967-07-03T`,
    'GovId' : 'LOGANB486CG',
    'GovIdType' : 'Driver License',
    'Address' : '43 Stockert Hollow Road, Everett, WA, 98203'
},
{
    'FirstName' : 'Alexis',
    'LastName' : 'Pena',
    'DOB' : `1974-02-10T`,
    'GovId' : '744 849 301',
    'GovIdType' : 'SSN',
    'Address' : '4058 Melrose Street, Spokane Valley, WA, 99206'
},
```

```
{
    'FirstName' : 'Melvin',
    'LastName' : 'Parker',
    'DOB' : `1976-05-22T`,
    'GovId' : 'P626-168-229-765',
    'GovIdType' : 'Passport',
    'Address' : '4362 Ryder Avenue, Seattle, WA, 98101'
},
{
    'FirstName' : 'Salvatore',
    'LastName' : 'Spencer',
    'DOB' : `1997-11-15T`,
    'GovId' : 'S152-780-97-415-0',
    'GovIdType' : 'Passport',
    'Address' : '4450 Honeysuckle Lane, Seattle, WA, 98101'
} >>
```

7. Füllen Sie dann die `DriversLicense`-Tabelle mit Dokumenten, die Führerschein-Informationen für jeden Fahrzeughalter enthalten.

   Geben Sie im Fenster des Abfrageeditors die folgende Anweisung ein und klicken Sie dann auf Run (Ausführen).

```
INSERT INTO DriversLicense
<< {
    'LicensePlateNumber' : 'LEWISR261LL',
    'LicenseType' : 'Learner',
    'ValidFromDate' : `2016-12-20T`,
    'ValidToDate' : `2020-11-15T`,
    'PersonId' : ''
},
{
    'LicensePlateNumber' : 'LOGANB486CG',
    'LicenseType' : 'Probationary',
    'ValidFromDate' : `2016-04-06T`,
    'ValidToDate' : `2020-11-15T`,
    'PersonId' : ''
},
{
    'LicensePlateNumber' : '744 849 301',
    'LicenseType' : 'Full',
    'ValidFromDate' : `2017-12-06T`,
    'ValidToDate' : `2022-10-15T`,
```

```
    'PersonId' : ''
},
{
    'LicensePlateNumber' : 'P626-168-229-765',
    'LicenseType' : 'Learner',
    'ValidFromDate' : `2017-08-16T`,
    'ValidToDate' : `2021-11-15T`,
    'PersonId' : ''
},
{
    'LicensePlateNumber' : 'S152-780-97-415-0',
    'LicenseType' : 'Probationary',
    'ValidFromDate' : `2015-08-15T`,
    'ValidToDate' : `2021-08-21T`,
    'PersonId' : ''
} >>
```

8. Jetzt füllen Sie die `VehicleRegistration`-Tabelle mit Dokumenten zur Fahrzeugzulassung. Diese Dokumente enthalten eine verschachtelte `Owners`-Struktur, in der die primären und sekundären Eigentümer gespeichert sind.

   Geben Sie im Fenster des Abfrageeditors die folgende Anweisung ein und klicken Sie dann auf Run (Ausführen).

```
INSERT INTO VehicleRegistration
<< {
    'VIN' : '1N4AL11D75C109151',
    'LicensePlateNumber' : 'LEWISR261LL',
    'State' : 'WA',
    'City' : 'Seattle',
    'PendingPenaltyTicketAmount' : 90.25,
    'ValidFromDate' : `2017-08-21T`,
    'ValidToDate' : `2020-05-11T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
},
{
    'VIN' : 'KM8SRDHF6EU074761',
    'LicensePlateNumber' : 'CA762X',
    'State' : 'WA',
    'City' : 'Kent',
```

```
    'PendingPenaltyTicketAmount' : 130.75,
    'ValidFromDate' : `2017-09-14T`,
    'ValidToDate' : `2020-06-25T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
},
{
    'VIN' : '3HGGK5G53FM761765',
    'LicensePlateNumber' : 'CD820Z',
    'State' : 'WA',
    'City' : 'Everett',
    'PendingPenaltyTicketAmount' : 442.30,
    'ValidFromDate' : `2011-03-17T`,
    'ValidToDate' : `2021-03-24T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
},
{
    'VIN' : '1HVBBAANXWH544237',
    'LicensePlateNumber' : 'LS477D',
    'State' : 'WA',
    'City' : 'Tacoma',
    'PendingPenaltyTicketAmount' : 42.20,
    'ValidFromDate' : `2011-10-26T`,
    'ValidToDate' : `2023-09-25T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
},
{
    'VIN' : '1C4RJFAG0FC625797',
    'LicensePlateNumber' : 'TH393F',
    'State' : 'WA',
    'City' : 'Olympia',
    'PendingPenaltyTicketAmount' : 30.45,
    'ValidFromDate' : `2013-09-02T`,
    'ValidToDate' : `2024-03-19T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
```

```
        'SecondaryOwners' : []
    }
} >>
```

9. Füllen Sie die `Vehicle`-Tabelle schließlich mit Dokumenten zur Beschreibung der Fahrzeuge, die in Ihrem Ledger registriert sind.

   Geben Sie im Fenster des Abfrageeditors die folgende Anweisung ein und klicken Sie dann auf Run (Ausführen).

```
INSERT INTO Vehicle
<< {
    'VIN' : '1N4AL11D75C109151',
    'Type' : 'Sedan',
    'Year' : 2011,
    'Make' : 'Audi',
    'Model' : 'A5',
    'Color' : 'Silver'
},
{
    'VIN' : 'KM8SRDHF6EU074761',
    'Type' : 'Sedan',
    'Year' : 2015,
    'Make' : 'Tesla',
    'Model' : 'Model S',
    'Color' : 'Blue'
},
{
    'VIN' : '3HGGK5G53FM761765',
    'Type' : 'Motorcycle',
    'Year' : 2011,
    'Make' : 'Ducati',
    'Model' : 'Monster 1200',
    'Color' : 'Yellow'
},
{
    'VIN' : '1HVBBAANXWH544237',
    'Type' : 'Semi',
    'Year' : 2009,
    'Make' : 'Ford',
    'Model' : 'F 150',
    'Color' : 'Black'
},
```

```
{
    'VIN' : '1C4RJFAG0FC625797',
    'Type' : 'Sedan',
    'Year' : 2019,
    'Make' : 'Mercedes',
    'Model' : 'CLK 350',
    'Color' : 'White'
} >>
```

Als Nächstes können Sie mit `SELECT`-Anweisungen Daten aus den Tabellen im `vehicle-registration`-Ledger lesen. Fahren Sie mit Schritt 3: Abfragen der Tabellen in einem Ledger fort.

Schritt 3: Abfragen der Tabellen in einem Ledger

Nachdem Sie Tabellen in einem Amazon QLDB-Ledger erstellt und sie mit Daten geladen haben, können Sie Abfragen ausführen, um die Fahrzeugregistrierungsdaten zu überprüfen, die Sie gerade eingegeben haben. QLDB verwendet PartiQL als Abfragesprache und Amazon Ion als dokumentorientiertes Datenmodell.

PartiQL ist eine quelloffene, SQL-kompatible Abfragesprache, die für die Arbeit mit Ion erweitert wurde. Mit PartiSQL können Sie Ihre Daten mit vertrauten SQL-Operatoren einfügen, abfragen und verwalten. Amazon Ion ist eine Obermenge von JSON. Ion ist ein dokumentenbasiertes Open-Source-Datenformat, das Ihnen die Flexibilität bietet, strukturierte, halbstrukturierte und verschachtelte Daten zu speichern und zu verarbeiten.

In diesem Schritt verwenden Sie `SELECT`-Anweisungen zum Lesen von Daten aus den Tabellen im `vehicle-registration`-Ledger.

> **Warning**
>
> Wenn Sie eine Abfrage in QLDB ohne indizierte Suche ausführen, wird ein vollständiger Tabellenscan aufgerufen. PartiQL unterstützt solche Abfragen, da es SQL-kompatibel ist. Führen Sie jedoch keine Tabellenscans für produktive Anwendungsfälle in QLDB aus. Tabellenscans können bei großen Tabellen zu Leistungsproblemen führen, einschließlich Parallelitätskonflikten und Transaktions-Timeouts.
> Um Tabellenscans zu vermeiden, müssen Sie Anweisungen mit einer`WHERE` Prädikatklausel ausführen, die einen Gleichheitsoperator für ein indiziertes Feld oder eine Dokument-ID

verwenden, z. B.`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`. Weitere Informationen finden Sie unter Optimieren der Abfrageleistung.

So fragen Sie die Tabellen ab

1. Öffnen Sie die Amazon QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich den PartiQL-Editor aus.
3. Wählen Sie den `vehicle-registration`-Ledger aus.
4. Geben Sie im Abfrage-Editor-Fenster die folgende Anweisung ein, um die Tabelle `Vehicle` nach einer bestimmten Fahrgestellnummer (Vehicle Identification Number, VIN) abzufragen, die Sie dem Ledger hinzugefügt haben. Wählen Sie dann Run (Ausführen) aus.

   Um die Anweisung auszuführen, können Sie auch die Tastenkombination Ctrl+Enter für Windows oder Cmd+Return für macOS verwenden. Weitere Tastenkombinationen finden Sie unter Tastenkombinationen für den PartiQL-Editor.

   ```
   SELECT * FROM Vehicle AS v
   WHERE v.VIN = '1N4AL11D75C109151'
   ```

5. Sie können innere Join-Abfragen schreiben. Dieses Abfragebeispiel verbindet `Vehicle` mit `VehicleRegistration` und gibt Zulassungsinformationen zusammen mit Attributen des registrierten Fahrzeugs für eine angegebene `VIN` zurück.

   Geben Sie die folgende Anweisung ein und klicken Sie dann auf Run (Ausführen).

   ```
   SELECT v.VIN, r.LicensePlateNumber, r.State, r.City, r.Owners
   FROM Vehicle AS v, VehicleRegistration AS r
   WHERE v.VIN = '1N4AL11D75C109151'
   AND v.VIN = r.VIN
   ```

   Sie können auch die `Person`- und `DriversLicense`-Tabellen verbinden, um Attribute in Bezug auf die Fahrer zu sehen, die zum Ledger hinzugefügt werden.

   Wiederholen Sie diesen Schritt für die folgenden Aussagen.

   ```
   SELECT * FROM Person AS p, DriversLicense AS l
   WHERE p.GovId = l.LicensePlateNumber
   ```

Weitere Informationen zum Ändern von Dokumenten in den Tabellen im `vehicle-registration`-Ledger finden Sie unter Schritt 4: Bearbeiten von Dokumenten in einem Ledger.

Schritt 4: Bearbeiten von Dokumenten in einem Ledger

Da Sie nun über Daten verfügen, mit denen Sie arbeiten können, können Sie damit beginnen, Änderungen an Dokumenten im`vehicle-registration` Hauptbuch in Amazon QLDB vorzunehmen. Betrachten Sie beispielsweise den Audi A5 mit VIN `1N4AL11D75C109151`. Dieses Auto war ursprünglich im Besitz eines Fahrers namens Raul Lewis in Seattle, WA.

Nehmen wir an, dass Raul das Auto an eine Person namens Brent Logan mit Wohnsitz in Everett, WA, verkauft. Anschließend entscheiden sich Brent und Alexis Pena zu heiraten. Brent möchte Alexis als sekundäre Eigentümer in der Zulassung hinzufügen. In diesem Schritt verdeutlichen die folgenden DML-Anweisungen, wie die entsprechenden Änderungen in Ihrem Ledger vorgenommen werden, um diese Ereignisse zu reflektieren.

> **Tip**
>
> Es hat sich bewährt, den einem Dokument vom System zugewiesenen Schlüssel`id` als Fremdschlüssel zu verwenden. Sie können zwar Felder definieren, die eindeutige Kennungen (z. B. eine Kfz-VIN) sein sollen, die echte eindeutige Kennung eines Dokuments ist aber seine `id`. Dieses Feld ist in den Metadaten des Dokuments enthalten, die Sie in der festgeschriebenen Ansicht (d. h. in der vom System definierten Ansicht der Tabelle) abrufen können.
> Weitere Informationen zu Ansichten in QLDB finden Sie unterSchlüsselkonzepte. Weitere Informationen zu Metadaten finden Sie unter Metadaten von Dokumenten abfragen.

So ändern Sie Dokumente

1. Öffnen Sie die Amazon QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich den PartiQL-Editor aus.
3. Wählen Sie den `vehicle-registration`-Ledger aus.

> **Note**
>
> Wenn Sie Ihr Ledger mithilfe der automatischen Funktion „Beispieldaten laden“ der Konsole einrichten, fahren Sie mit Schritt 6 fort.

4. Wenn Sie `INSERT`-Anweisungen zum Laden der Beispieldaten manuell ausgeführt haben, fahren Sie mit diesen Schritten fort.

   Um Raul zuerst als Eigentümer des Fahrzeugs zu registrieren, beginnen Sie damit, indem Sie sein vom System zugewiesenes Dokument `id` in der `Person`-Tabelle suchen. Dieses Feld ist in den Metadaten des Dokuments enthalten, die Sie in der vom System definierten Ansicht der Tabelle abrufen können. Diese wird bestätigte Ansicht genannt.

   Geben Sie im Fenster des Abfrageeditors die folgende Anweisung ein und klicken Sie dann auf Run (Ausführen).

   ```
   SELECT metadata.id FROM _ql_committed_Person AS p
   WHERE p.data.FirstName = 'Raul' and p.data.LastName = 'Lewis'
   ```

   Das Präfix `_ql_committed_` ist ein reserviertes Präfix, das angibt, dass Sie die bestätigte Ansicht der `Person`-Tabelle abfragen möchten. In dieser Ansicht sind Ihre Daten im `data`-Feld verschachtelt und Metadaten sind im `metadata`-Feld verschachtelt.

5. Verwenden Sie jetzt diese `id` in einer `UPDATE`-Anweisung, um das entsprechende Dokument in der `VehicleRegistration`-Tabelle zu ändern. Geben Sie die folgende Anweisung ein und klicken Sie dann auf Run (Ausführen).

   ```
   UPDATE VehicleRegistration AS r
   SET r.Owners.PrimaryOwner.PersonId = '294jJ3YUoH1IEEm8GSabOs' --replace with your
    id
   WHERE r.VIN = '1N4AL11D75C109151'
   ```

   Vergewissern Sie sich, dass Sie das `Owners`-Feld geändert haben, indem Sie diese Anweisung ausgeben.

   ```
   SELECT r.Owners FROM VehicleRegistration AS r
   WHERE r.VIN = '1N4AL11D75C109151'
   ```

6. Zum Übertragen der Eigentümerschaft des Fahrzeugs an Brent in der Stadt Everett suchen Sie zunächst seine `id` in der `Person`-Tabelle mit der folgenden Anweisung.

```
SELECT metadata.id FROM _ql_committed_Person AS p
WHERE p.data.FirstName = 'Brent' and p.data.LastName = 'Logan'
```

Verwenden Sie anschließend diese `id` zum Aktualisieren des `PrimaryOwner` und des `City` in der `VehicleRegistration`-Tabelle.

```
UPDATE VehicleRegistration AS r
SET r.Owners.PrimaryOwner.PersonId = '7NmE8YLPbXc0IqesJy1rpR', --replace with your
 id
    r.City = 'Everett'
WHERE r.VIN = '1N4AL11D75C109151'
```

Vergewissern Sie sich, dass Sie die `City`- und `PrimaryOwner`-Felder durch Ausgabe dieser Anweisung geändert haben.

```
SELECT r.Owners.PrimaryOwner, r.City
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
```

7. Um Alexis als sekundäre Eigentümerin des Fahrzeugs hinzuzufügen, suchen Sie ihre `Person` `id`.

```
SELECT metadata.id FROM _ql_committed_Person AS p
WHERE p.data.FirstName = 'Alexis' and p.data.LastName = 'Pena'
```

Fügen Sie diese `id` anschließend in die `SecondaryOwners`-Liste mit der folgenden FROM-INSERT-DML-Anweisung hinzu.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners
    VALUE { 'PersonId' : '5Ufgdlnj06gF5CWc0Iu64s' } --replace with your id
```

Vergewissern Sie sich, dass Sie `SecondaryOwners` geändert haben, indem Sie diese Anweisung ausgeben.

```
SELECT r.Owners.SecondaryOwners FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
```

Informationen zum Überprüfen dieser Änderungen im `vehicle-registration`-Ledger finden Sie unter Schritt 5: Anzeigen des Revisionsverlaufs für ein Dokument.

Schritt 5: Anzeigen des Revisionsverlaufs für ein Dokument

Nach dem Bearbeiten der Zulassungsdaten für das Fahrzeug mit VIN `1N4AL11D75C109151` können Sie den Verlauf aller registrierten Eigentümer und alle anderen aktualisierten Felder abfragen. Sie können alle Revisionen eines Dokuments anzeigen, die Sie eingefügt, aktualisiert und gelöscht haben, indem Sie die integrierte Verlaufsfunktion abfragen.

Die Verlaufsfunktion gibt Revisionen aus der festgeschriebenen Ansicht Ihrer Tabelle zurück, die sowohl Ihre Anwendungsdaten als auch die zugehörigen Metadaten enthält. Die Metadaten zeigen genau, wann und in welcher Reihenfolge die jeweiligen Revisionen gemacht wurden und durch welche Transaktion sie bestätigt wurden.

In diesem Schritt führen Sie eine Abfrage des Änderungsverlaufs eines Dokuments in der `VehicleRegistration`-Tabelle im `vehicle-registration`-Ledger durch.

So zeigen Sie den Revisionsverlauf an

1. Öffnen Sie die Amazon QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich den PartiQL-Editor aus.
3. Wählen Sie den `vehicle-registration`-Ledger aus.
4. Um den Verlauf eines Dokuments abzufragen, beginnen Sie mit der Ermittlung der eindeutigen `id`. Zusätzlich zum Abfragen der bestätigten Ansicht ist eine weitere Möglichkeit zum Abrufen einer Dokument-`id` die Verwendung des `BY`-Schlüsselworts in der Standard-Benutzeransicht der Tabelle. Weitere Informationen hierzu finden Sie unter Verwenden der BY-Klausel zur Abfrage der Dokument-ID.

   Geben Sie im Fenster des Abfrageeditors die folgende Anweisung ein und klicken Sie dann auf Run (Ausführen).

   ```
   SELECT r_id FROM VehicleRegistration AS r BY r_id
   ```

```
WHERE r.VIN = '1N4AL11D75C109151'
```

5. Als Nächstes können Sie diesen`id` Wert verwenden, um die Verlaufsfunktion abzufragen. Geben Sie die folgende Anweisung ein und klicken Sie dann auf Run (Ausführen). Achten Sie darauf, den `id`-Wert ggf. durch Ihre eigene Dokument-ID zu ersetzen.

```
SELECT h.data.VIN, h.data.City, h.data.Owners
FROM history(VehicleRegistration) AS h
WHERE h.metadata.id = 'ADR2LQq48kB9neZDupQrMm' --replace with your id
```

> **Note**
>
> Für die Zwecke dieses Tutorials gibt diese Protokollabfrage alle Versionen der Dokument-ID zurück`ADR2LQq48kB9neZDupQrMm`. Es hat sich jedoch bewährt, eine Verlaufsabfrage sowohl mit einer Dokument-ID als auch mit einem Datumsbereich (Startzeit und Endzeit) zu qualifizieren.
> In QLDB wird jede`SELECT` Anfrage in einer Transaktion verarbeitet und unterliegt einem Transaktions-Timeout-Limit. Verlaufsabfragen, die eine Start- und Endzeit enthalten, profitieren von der Qualifizierung des Datumsbereichs. Weitere Informationen finden Sie unter Verlaufsfunktion.

Die Historienfunktion gibt Dokumente im selben Schema wie die Committed-Ansicht zurück. In diesem Beispiel werden Ihre geänderten Fahrzeugregisterdaten projiziert. Die Ausgabe sollte in etwa folgendermaßen aussehen:

VIN	Ort	Eigentümer
`"1N4AL11D 75C109151"`	`"Seattle"`	`{PrimaryOwner:{PersonId:""} ,SecondaryOwners:[]}`
`"1N4AL11D 75C109151"`	`"Seattle"`	`{PrimaryOwner:{PersonId:"29 4jJ3YUoH1IEEm8GSabOs"}, SecondaryOwners:[]}`

VIN	Ort	Eigentümer
"1N4AL11D 75C109151"	"Everett"	{PrimaryOwner:{PersonId:"7N mE8YLPbXc0IqesJy1rpR"}, SecondaryOwners:[]}
"1N4AL11D 75C109151"	"Everett"	{PrimaryOwner:{PersonId:"7N mE8YLPbXc0IqesJy1rpR"}, SecondaryOwners:[{PersonId: "5Ufgdlnj06gF5CWcOIu64s"}]}

> **Note**
>
> Der Abfrageverlauf gibt die Dokumentrevisionen möglicherweise nicht immer in sequenzieller Reihenfolge zurück.

Prüfen Sie die Ausgabe und vergewissern Sie sich, dass die Änderungen widerspiegeln, was Sie in Schritt 4: Bearbeiten von Dokumenten in einem Ledger durchgeführt haben.

6. Anschließend können Sie die Dokumentmetadaten der einzelnen Revision näher untersuchen. Geben Sie die folgende Anweisung ein und klicken Sie dann auf Run (Ausführen). Stellen Sie auch hier sicher, dass Sie den `id`-Wert durch Ihre eigene Dokument-ID ersetzen.

```
SELECT VALUE h.metadata
FROM history(VehicleRegistration) AS h
WHERE h.metadata.id = 'ADR2LQq48kB9neZDupQrMm' --replace with your id
```

Die Ausgabe sollte in etwa folgendermaßen aussehen:

Versic	ID	txTime	txId
0	"ADR2LQq48kB9neZDu pQrMm"	2019-05-23T19:20:3 60d-3Z	"FMoVdWuPxJg3k466I z4i75"
1	"ADR2LQq48kB9neZDu pQrMm"	2019-05-23T21:40:1 99d-3Z	"KWByxe842Xw8DNHcv ARPOt"

Versic	ID	txTime	txId
2	"ADR2LQq48kB9neZDupQrMm"	2019-05-23T21:44:432d-3Z	"EKwDOJRwbHpFvmAyJ2Kdh9"
3	"ADR2LQq48kB9neZDupQrMm"	2019-05-23T21:49:254d-3Z	"96EiZd7vCmJ6RAvOvTZ4YA"

Diese Metadatenfelder bieten Informationen dazu, wann die einzelnen Elemente geändert wurden und welcher Transaktion. Aus diesen Daten können Sie Folgendes schließen:

- Das Dokument wird eindeutig durch seine vom System zugewiesene `id` identifiziert: `ADR2LQq48kB9neZDupQrMm`. Dies ist eine universell eindeutige Kennung (UUID), die in einer Base62-kodierten Zeichenfolge dargestellt wird.
- Die `txTime` zeigt, dass die ursprüngliche Revision des Dokuments (Version `0`) am `2019-05-23T19:20:360d-3Z` erstellt wurde.
- Jede nachfolgende Transaktion erstellt eine neue Revision mit derselben Dokument-`id`, eine aufsteigenden Versionsnummer und einer aktualisierten `txId` und `txTime`.

Um die Revision eines Dokuments im`vehicle-registration` Ledger kryptografisch zu überprüfen, fahren Sie mit fortSchritt 6: Überprüfen eines Dokuments in einem Ledger.

Schritt 6: Überprüfen eines Dokuments in einem Ledger

Mit Amazon QLDB können Sie die Integrität eines Dokuments im Journal Ihres Hauptbuchs effizient überprüfen, indem Sie kryptografisches Hashing mit SHA-256 verwenden. In diesem Beispiel entscheiden sich Alexis und Brent, auf ein neues Modell aufzurüsten, indem sie das Fahrzeug mit VIN `1N4AL11D75C109151` bei einem Autohändler in Zahlung geben. Der Vertragshändler startet den Prozess, indem er die Eigentümerschaft des Fahrzeugs bei der Zulassungsstelle überprüft.

Weitere Informationen zur Funktionsweise von Verifizierung und kryptografischem Hashing in QLDB finden Sie unterDatenverifizierung in Amazon QLDB.

In diesem Schritt überprüfen Sie eine Dokumentreversion im `vehicle-registration`-Ledger. Zuerst fordern Sie einen Digest an, der als Ausgabedatei zurückgegeben wird und als Signatur des gesamten Änderungsverlaufs Ihres Ledgers fungiert. Anschließend fordern Sie einen Nachweis für

die Revision in Bezug auf diesen Digest an. Mit diesem Nachweis wird die Integrität Ihrer Revision verifiziert, wenn alle Validierungsprüfungen bestanden werden.

So fordern Sie ein Digest an

1. Öffnen Sie die Amazon QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich Ledgers aus.
3. Wählen Sie in der Liste der Ledgers `vehicle-registration` aus.
4. Wählen Sie Get digest (Digest abrufen). Dasm Dialogfeld Get digest (Digest abrufen) zeigt die folgenden Digest-Details:

   - Digest — Der SHA-256-Hashwert des Digest, den Sie angefordert haben.
   - Adresse des Übersichtstips — Die letzte Blockposition im Journal, die in der von Ihnen angeforderten Zusammenfassung enthalten ist. Eine Adresse hat die folgenden zwei Felder:
     - `strandId`— Die eindeutige ID des Journalstrangs, der den Block enthält.
     - `sequenceNo`— Die Indexnummer, die die Position des Blocks innerhalb des Strangs angibt.
   - Ledger — Der Name des Buchs, für das Sie eine Zusammenfassung angefordert haben.
   - Datum — Der Zeitstempel, zu dem Sie den Digest angefordert haben.
5. Überprüfen Sie die Digest-Informationen. Wählen Sie dann Speichern. Sie können den Standard-Dateinamen behalten oder einen neuen Namen eingeben.

   Dieser Schritt speichert eine Klartext-Datei mit Inhalten im Amazon Ion-Format. Die Datei verfügt über die Dateinamenerweiterung `.ion.txt` und enthält alle Digest-Informationen, die im vorherigen Dialogfeld aufgelistet wurden. Im Folgenden finden Sie ein Beispiel für die Inhalte einer Digest-Datei. Die Reihenfolge der Felder kann je nach Browser variieren.

   ```
   {
     "digest": "42zaJOfV8iGutVGNaIuzQWhD5Xb/5B9lScHnvxPXm9E=",
     "digestTipAddress": "{strandId:\"BlFTjlSXze9BIh1KOszcE3\",sequenceNo:73}",
     "ledger": "vehicle-registration",
     "date": "2019-04-17T16:57:26.749Z"
   }
   ```
6. Speichern Sie diese Datei an einem Ort, an dem Sie später darauf zugreifen können. In den folgenden Schritten verwenden Sie diese Datei, um den Vergleich einer Dokumentversion zu überprüfen.

Nachdem Sie ein Ledger-Digest gespeichert haben, können Sie eine Dokumentrevision in Bezug auf diesen Digest überprüfen.

> **Note**
>
> In einem produktiven Anwendungsfall zur Überprüfung verwenden Sie eine Übersicht, die zuvor gespeichert wurde, anstatt die beiden Aufgaben nacheinander auszuführen. Es hat sich bewährt, die Zusammenfassung anzufordern und zu speichern, sobald eine Revision, die Sie später überprüfen möchten, in das Journal geschrieben wird.

So überprüfen Sie eine Dokumentrevision

1. Führen Sie zunächst eine Abfrage Ihres Ledgers für die `id` und die `blockAddress` der Dokumentrevision durch, die Sie verifizieren möchten. Diese Felder sind in den Metadaten des Dokuments enthalten. Sie können diese Abfrage in der bestätigten Ansicht vornehmen.

   Das Dokument`id` ist eine vom System zugewiesene eindeutige ID-Zeichenfolge. Das`blockAddress` ist eine Ionen-Struktur, die die Blockposition angibt, an der die Revision übernommen wurde.

   Wählen Sie im Navigationsbereich der QLDB-Konsole aus.
2. Wählen Sie den `vehicle-registration`-Ledger aus.
3. Geben Sie im Fenster des Abfrageeditors die folgende Anweisung ein und klicken Sie dann auf Run (Ausführen).

   ```
   SELECT r.metadata.id, r.blockAddress
   FROM _ql_committed_VehicleRegistration AS r
   WHERE r.data.VIN = '1N4AL11D75C109151'
   ```
4. Kopieren und speichern Sie die `id`- und `blockAddress`-Werte, die Ihre Abfrage zurückgibt. Achten Sie darauf, dass Sie die doppelten Anführungszeichen für das `id`-Feld auslassen. In Amazon Ion werden Zeichenfolge-Datentypen in doppelte Anführungszeichen gesetzt.
5. Nachdem Sie eine Dokumentrevision ausgewählt haben, können Sie damit beginnen, sie zu überprüfen.

   Wählen Sie im Navigationsbereich die Option Verification (Überprüfung) aus.

6. Geben Sie auf dem Formular Verify document (Dokument überprüfen) unter Specify the document that you want to verify (Das Dokument angeben, das Sie überprüfen möchten) die folgenden Eingabeparameter ein:

   - Ledger — Wähle`vehicle-registration`.
   - Adresse blockieren — Der`blockAddress` Wert, den Ihre Abfrage in Schritt 3 zurückgegeben hat.
   - Dokument-ID — Der`id` Wert, den Ihre Abfrage in Schritt 3 zurückgegeben hat.

7. Wählen Sie unter Specify the document that you want to verify (Das Dokument angeben, das Sie überprüfen möchten) den Digest aus, den Sie zuvor gespeichert haben, indem Sie Choose digest (Digest auswählen) auswählen. Wenn die Datei gültig ist, werden alle Digest-Felder auf Ihrer Konsole automatisch gefüllt. Sie können die folgenden Werte auch manuell direkt aus Ihrer Digest-Datei kopieren und einfügen:

   - Digest — Der`digest` Wert aus Ihrer Digest-Datei.
   - Adresse des Übersichtstips — Der`digestTipAddress` Wert aus Ihrer Übersichtsdatei.

8. Überprüfen Sie Ihre Dokument- und Digest-Eingabeparameter und wählen Sie anschließend Verify (Überprüfen) aus.

   Die Konsole automatisiert zwei Schritte für Sie:

   a. Fordern Sie von QLDB einen Nachweis für Ihr spezifiziertes Dokument an.

   b. Verwenden Sie den von QLDB zurückgegebenen Nachweis, um eine clientseitige API aufzurufen, die die Revision Ihres Dokuments anhand der bereitgestellten Übersicht überprüft.

   Die Konsole zeigt die Ergebnisse Ihrer Anforderung in der Karte Verification results (Überprüfungsergebnisse) an. Weitere Informationen finden Sie unter Verifizierungsergebnisse.

9. Um die Überprüfungslogik zu testen, wiederholen Sie die Schritte 6—8 unter So überprüfen Sie eine Dokumentversion, ändern Sie jedoch ein einzelnes Zeichen in der Digest-Eingabezeichenfolge. Dies sollte dazu führen, dass Ihre Verify-Anforderung (Überprüfungsanforderung) fehlschlägt und eine entsprechende Fehlermeldung ausgegeben wird.

Wenn Sie den `vehicle-registration`-Ledger nicht mehr verwenden müssen, fahren Sie mit Schritt 7 (optional): Bereinigen von Ressourcen fort.

Schritt 7 (optional): Bereinigen von Ressourcen

Sie können den `vehicle-registration`-Ledger weiterhin verwenden. Wenn Sie ihn nicht mehr benötigen, sollten Sie ihn jedoch löschen.

Wenn der Löschschutz für Ihr Ledger aktiviert ist, müssen Sie ihn zuerst deaktivieren, bevor Sie das Ledger mithilfe der QLDB-API,AWS Command Line Interface (AWS CLI) oder der QLDB-Konsole löschen können.

So löschen Sie den Ledger

1. Öffnen Sie die Amazon QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich Ledgers aus.
3. Wenn der Löschschutz für dieses Ledger aktiviert ist, müssen Sie ihn zuerst deaktivieren.

   Wählen Sie in der Liste der Bücher die Option Buch bearbeiten aus`vehicle-registration`, und wählen Sie dann die Option Buch bearbeiten aus.
4. Schalten Sie auf der Seite „Buch bearbeiten“ den Löschschutz aus und wählen Sie dann „Änderungen bestätigen“.
5. Wählen Sie in der Liste der Bücher`vehicle-registration` erneut aus, und wählen Sie dann Löschen.
6. Bestätigen Sie dies, indem Sie **`delete vehicle-registration`** in das bereitgestellte Feld eingeben.

Weitere Informationen zur Arbeit mit Ledgers in QLDB finden Sie unterErste Schritte mit Amazon QLDB: Nächste Schritte.

Erste Schritte mit Amazon QLDB: Nächste Schritte

Weitere Informationen zur Verwendung von Amazon QLDB finden Sie in den folgenden Themen:

- Arbeiten mit Daten und Historie in Amazon QLDB
- Erste Schritte mit dem Amazon-QLDB-Treiber
- Amazon QLDB-Nebenläufigkeit von Amazon QLDB-Nebenläufigkeit
- Exportieren von Journaldaten aus Amazon QLDB
- Streaming von Journaldaten aus Amazon QLDB

- Datenverifizierung in Amazon QLDB
- Amazon QLDB PartiQL-Referenz

Erste Schritte mit dem Amazon-QLDB-Treiber

Dieses Kapitel enthält praktische Tutorials, die Ihnen helfen, die Entwicklung mit Amazon QLDB mithilfe des QLDB-Treiber zu erlernen. Der Treiber basiert auf demAWS SDK, das die Interaktion mit der QLDB-API unterstützt.

QLDB-Sitzungsabstraktion

Der Treiber stellt eine abstrakte Ebene auf hoher Ebene über der Transaktionsdaten-API (QLDB-Sitzung) bereit. Es optimiert den Prozess der Ausführung von PartiQL-Anweisungen auf Ledger-Daten, indem es SendCommandAPI-Aufrufe verwaltet. Für diese API-Aufrufe sind mehrere Parameter erforderlich, die der Treiber für Sie verarbeitet, einschließlich der Verwaltung von Sitzungen, Transaktionen und der Wiederholungsrichtlinie im Falle von Fehlern. Der Treiber verfügt außerdem über Leistungsoptimierungen und wendet bewährte Methoden für die Interaktion mit QLDB an.

> **Note**
>
> Um mit den Ressourcenmanagement-API-Vorgängen zu interagieren, die in der Amazon QLDB-API-Referenz aufgeführt sind, verwenden Sie dasAWS SDK direkt anstelle des Treibers. Sie verwenden die Management-API nur für die Verwaltung von Ledger-Ressourcen und für nicht transaktionale Datenvorgänge wie Export, Streaming und Datenüberprüfung.

Amazon Ion-Unterstützung

Darüber hinaus verwendet der Treiber Amazon Ion-Bibliotheken, um die Verarbeitung von Ion-Daten bei der Ausführung von Transaktionen zu unterstützen. Diese Bibliotheken kümmern sich auch um die Berechnung des Hashs der Ionen-Werte. QLDB benötigt diese Ionen-Hashes, um die Integrität von Datentransaktionsanforderungen zu überprüfen.

Fahrerterminologie

Dieses Tool wird als Treiber bezeichnet, da es mit anderen Datenbanktreibern vergleichbar ist, die entwicklerfreundliche Schnittstellen bieten. Diese Treiber kapseln in ähnlicher Weise Logik, die einen Standardsatz von Befehlen und Funktionen in spezifische Aufrufe umwandelt, die für die Low-Level-API des Dienstes erforderlich sind.

Der Treiber ist Open Source GitHub und für die folgenden Programmiersprachen verfügbar:

- Java-Treiber
- .NET-Treiber
- Wechseln Sie zum Fahrer.
- Treiber Node.js
- Python-Treiber

Allgemeine Treiberinformationen für alle unterstützten Programmiersprachen sowie zusätzliche Tutorials finden Sie in den folgenden Themen:

- Sitzungsmanagement mit dem Fahrer
- Treiberempfehlungen
- Richtlinie für die Wiederholung von T
- Häufige Fehler
- Bereitstellung einer Beispielanwendung
- Arbeiten mit
- Abrufen von PartiQL-Anweisungsstatistiken

Amazon QLDB-Treiber für Java

Um mit Daten in Ihrem Ledger zu arbeiten, können Sie von Ihrer Java-Anwendung aus mithilfe einesAWS bereitgestellten Treibers eine Verbindung zu Amazon QLDB herstellen. In den folgenden Themen wird die erste Schritte mit dem QLDB-Treiber für Java beschrieben.

Themen

- Ressourcen für Fahrer
- Voraussetzungen
- Festlegen IhrerAWS Standardanmeldeinformationen und Ihrer Region
- Installation
- Amazon QLDB-Treiber für Java — Schnellstart-Tutorial
- Amazon QLDB-Treiber für Java — Kochbuch-Referenz

Ressourcen für Fahrer

Weitere Informationen zu den vom Java-Treiber unterstützten Funktionen finden Sie in den folgenden Ressourcen:

- API-Referenz: 2.x, 1.x
- Treiber-Quellcode (GitHub)
- Quellcode der Beispielanwendung (GitHub)
- Ledger-Loader-Framework (GitHub)
- Beispiele für Amazon Ion-Code

Voraussetzungen

Bevor Sie mit dem QLDB-Treiber für Java beginnen, müssen Sie Folgendes tun:

1. Befolgen Sie die AWS-Einrichtungsanweisungen unter Zugreifen auf Amazon QLDB. Diese umfasst die folgenden Funktionen:
    1. Registrieren Sie sich fürAWS.
    2. Erstellen Sie sich einen Benutzer mit den entsprechenden QLDB Berechtigungen.
    3. Erteilen programmgesteuerten Zugriffs
2. Richten Sie eine Java-Entwicklungsumgebung ein, indem Sie Folgendes herunterladen und installieren:
    1. Java SE Development Kit 8, z. B. Amazon Corretto 8.
    2. (Optional) Integrierte Java-Entwicklungsumgebung (IDE) Ihrer Wahl, wie Eclipse oder IntelliJ.
3. Konfigurieren Sie Ihre Entwicklungsumgebung für dasAWS SDK for Java byFestlegen IhrerAWS Standardanmeldeinformationen und Ihrer Region.

Als Nächstes können Sie die vollständige Beispielanwendung für das Tutorial herunterladen — oder Sie können nur den Treiber in einem Java-Projekt installieren und kurze Codebeispiele ausführen.

- Um den QLDB-Treiber und denAWS SDK for Java in einem vorhandenen Projekt zu installieren, fahren Sie mit fortInstallation.

- Informationen zum Einrichten eines Projekts und zur Ausführung von kurzen Codebeispielen, die grundlegende Datentransaktionen in einem Ledger veranschaulichen, finden Sie unterErstErstErstErstErste-Erst.
- Ausführlichere Beispiele für Daten- und Verwaltungs-API-Operationen finden Sie in der vollständigen Beispielanwendung des Tutorials unterJava-Anleitung.

Festlegen IhrerAWS Standardanmeldeinformationen und Ihrer Region

Der QLDB-Treiber und die zugrunde liegenden Komponenten AWS SDK for Javasetzen voraus, dass SieAWS -Anmeldeinformationen für Ihre Anwendung zur Laufzeit bereitstellen. In den Codebeispielen in diesem Leitfaden wird davon ausgegangen, dass Sie eineAWS -Anmeldeinformationsdatei verwenden, wie unter Einrichten Standardeinformationen und Region im AWS SDK for Java 2.x-Entwicklerhandbuch beschrieben.

Im Rahmen dieser Schritte sollten Sie auch Ihre Standardeinstellung festlegen,AWS-Region um Ihren Standard-QLDB-Endpunkt zu bestimmen. Die Codebeispiele stellen standardmäßig eine Verbindung zu QLDB herAWS-Region. Eine vollständige Liste der Regionen, in denen QLDB verfügbar ist, finden Sie unter Amazon QLDB-Endpunkte und Kontingente in der Allgemeine AWS-Referenz.

Im Folgenden finden Sie ein Beispiel für eine AWS-Anmeldeinformationsdatei mit dem Namen `~/.aws/credentials`, wobei das Tildezeichen (~) das Stammverzeichnis repräsentiert.

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

Ersetzen Sie die Werte *`your_access_key_id und your_secret_access_key`* durch Ihre eigenenAWS Anmeldeinformationen.

Installation

QLDB unterstützt die folgenden Java-Treiberversionen und ihreAWS SDK-Abhängigkeiten.

Treiberversion	AWS-SDK	Status	Datum der letzten Veröffentlichung
1.x	AWS SDK for Java1.x	Produktionsfreigabe	20. März 2020

Treiberversion	AWS-SDK	Status	Datum der letzten Veröffentlichung
2.x	AWS SDK for Java 2.x	Produktionsfreigabe	4. Juni 2021

Um den QLDB-Treiber zu installieren, empfehlen wir die Verwendung eines Abhängigkeitsverwaltungssystems wie Gradle oder Maven. Fügen Sie beispielsweise das folgende Artefakt als Abhängigkeit in Ihrem Java-Projekt hinzu.

2.x

Gradle

Fügen Sie diese Abhängigkeit in Ihre`build.gradle` Konfigurationsdatei ein.

```
dependencies {
    compile group: 'software.amazon.qldb', name: 'amazon-qldb-driver-java', version:
 '2.3.1'
}
```

Maven

Fügen Sie diese Abhängigkeit in Ihre`pom.xml` Konfigurationsdatei ein.

```
<dependencies>
  <dependency>
    <groupId>software.amazon.qldb</groupId>
    <artifactId>amazon-qldb-driver-java</artifactId>
    <version>2.3.1</version>
  </dependency>
</dependencies>
```

Dieses Artefakt enthält automatisch das AWS SDK for Java 2.x-Core-Modul, Amazon Ion-Bibliotheken und andere erforderliche Abhängigkeiten.

1.x

Gradle

Fügen Sie diese Abhängigkeit in Ihre`build.gradle` Konfigurationsdatei ein.

```
dependencies {
    compile group: 'software.amazon.qldb', name: 'amazon-qldb-driver-java', version:
 '1.1.0'
}
```

Maven

Fügen Sie diese Abhängigkeit in Ihrepom.xml Konfigurationsdatei ein.

```
<dependencies>
  <dependency>
    <groupId>software.amazon.qldb</groupId>
    <artifactId>amazon-qldb-driver-java</artifactId>
    <version>1.1.0</version>
  </dependency>
</dependencies>
```

Dieses Artefakt enthält automatisch das AWS SDK for Java-Core-Modul, Amazon Ion-Bibliotheken und andere erforderliche Abhängigkeiten.

> **Important**
>
> Amazon Ion-Namespace — Wenn Sie Amazon Ion-Klassen in Ihre Anwendung importieren, müssen Sie das Paket verwenden, das sich unter dem Namespace befindetcom.amazon.ion. DasAWS SDK for Java hängt von einem anderen Ion-Paket unter dem Namespace absoftware.amazon.ion, aber dies ist ein Legacy-Paket, das nicht mit dem QLDB-Treiber kompatibel ist.

Kurze Codebeispiele für die Ausführung grundlegender Datentransaktionen in einem Ledger finden Sie unterKochbuchreferenz.

Andere optionale Bibliotheken

Optional können Sie auch die folgenden nützlichen Bibliotheken zu Ihrem Projekt hinzufügen. Diese Artefakte sind erforderliche Abhängigkeiten in derJava-Anleitung Beispielanwendung.

1. aws-java-sdk-qldb— Das QLDB-Modul derAWS SDK for Java. Die unterstützte Mindestversion von QLDB ist1.11.785.

Verwenden Sie dieses Modul in Ihrer Anwendung, um direkt mit den Management-API-Vorgängen zu interagieren, die in der aufgeführt sindAmazon QLDB API-Referenz.

2. jackson-dataformat-ion— Das Jackson-Datenformatmodul von FasterXML für Ion. Die Beispielanwendung erfordert Version `2.10.0` oder höher.

Gradle

Fügen Sie diese Abhängigkeiten in Ihre`build.gradle` Konfigurationsdatei ein.

```
dependencies {
    compile group: 'com.amazonaws', name: 'aws-java-sdk-qldb', version: '1.11.785'
    compile group: 'com.fasterxml.jackson.dataformat', name: 'jackson-dataformat-
ion', version: '2.10.0'
}
```

Maven

Fügen Sie diese Abhängigkeiten in Ihre`pom.xml` Konfigurationsdatei ein.

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-qldb</artifactId>
    <version>1.11.785</version>
  </dependency>
  <dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-ion</artifactId>
    <version>2.10.0</version>
  </dependency>
</dependencies>
```

Amazon QLDB-Treiber für Java — Schnellstart-Tutorial

In diesem -Tutortetetetetetetete-Tutortetetetetetetetetete-Tutortetetetetetetetetete-Tutortetetetetetetetete-Tutortetetetete Dieses Handbuch enthält Schritte zum Installieren des Treibers und kurze Codebeispiele für grundlegende CRUD-Vorgänge (Create, Read, Update und Delete). Ausführlichere Beispiele, die diese Vorgänge in einer vollständigen Beispielanwendung veranschaulichen, finden Sie im Java-Anleitung.

Themen

Voraussetzungen

Bevor Sie beginnen, stellen Sie sicher, dass die folgende Voraussetzung erfüllt ist:

1. Vervollständigen Sie dasVoraussetzungen für den Java-Treiber, falls noch nicht geschehen, für den Java-Treiber, falls noch nicht geschehen. Dazu gehören die RegistrierungAWS, die Gewährung von programmatischem Zugriff für die Entwicklung und die Installation einer integrierten Java-Entwicklungsumgebung (IDE).
2. Ledger mit dem Namen `quick-start` erstellen.

   Informationen zum Erstellen eines Ledgers finden Sie unterGrundfunktionen für Amazon QLDB-Ledgers oderSchritt 1: Erstellen eines neuen Ledgers unter Erste Schritte mit der Konsole.

Schritt 1: Einrichten des Projekts

Zuerst richten Sie Ihr Java-Projekt ein. Wir empfehlen, für dieses Tutorial das Maven-Abhängigkeitsverwaltungssystem zu verwenden.

> **Note**
>
> Wenn Sie eine IDE verwenden, die Funktionen zur Automatisierung dieser Einrichtungsschritte enthält, können Sie mit dem Vorgang fortfahrenSchritt 2: Initialisieren des Treibers.

1. Erstellen Sie einen Ordner für Ihre Anwendung.

```
$ mkdir myproject
$ cd myproject
```

2. Geben Sie Sie Sie den folgenden Befehl in Ihrem Terminal, um Ihr Projekt von einer Maven-Vorlage aus zu initialisieren. Ersetzen Sie *das Projektpaket, den Projektnamen* und die *Maven-Vorlage* gegebenenfalls durch Ihre eigenen Werte.

```
$ mvn archetype:generate
  -DgroupId=project-package \
  -DartifactId=project-name \
  -DarchetypeArtifactId=maven-template \
  -DinteractiveMode=false
```

   Für das *Maven-Template* können Sie das grundlegende Maven-Template verwenden:`maven-archetype-quickstart`

3. Um den QLDB-Treiber für Java als Projektabhängigkeit hinzuzufügen, navigieren Sie zu Ihrer neu erstelltenpom.xml Datei und fügen Sie das folgende Artefakt hinzu.

```
<dependency>
    <groupId>software.amazon.qldb</groupId>
    <artifactId>amazon-qldb-driver-java</artifactId>
    <version>2.3.1</version>
</dependency>
```

   Dieses Artefakt enthält automatisch das AWS SDK for Java 2.x-Core-Modul, Amazon Ion-Bibliotheken und andere erforderliche Abhängigkeiten. Ihrepom.xml Datei sollte jetzt in etwa folgendermaßen aussehen:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>software.amazon.qldb</groupId>
  <artifactId>qldb-quickstart</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>qldb-quickstart</name>
  <url>http://maven.apache.org</url>
```

```
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>software.amazon.qldb</groupId>
        <artifactId>amazon-qldb-driver-java</artifactId>
        <version>2.3.1</version>
    </dependency>
  </dependencies>
</project>
```

4. Öffnen Sie die `App.java` Datei.

   Fügen Sie dann schrittweise die Codebeispiele in den folgenden Schritten hinzu, um einige grundlegende CRUD-Operationen auszuprobieren. Oder Sie können das step-by-step Tutorial überspringen und stattdessen die komplette Anwendung ausführen.

Schritt 2: Initialisieren des Treibers

Initialisieren Sie eine Instance des Treibers, der eine Verbindung mit dem Ledger `quick-start` herstellt. Fügen Sie nun folgenden Code in die Datei `App.java` ein:

```
import java.util.*;
import com.amazon.ion.*;
import com.amazon.ion.system.*;
import software.amazon.awssdk.services.qldbsession.QldbSessionClient;
import software.amazon.qldb.*;

public final class App {
    public static IonSystem ionSys = IonSystemBuilder.standard().build();
    public static QldbDriver qldbDriver;

    public static void main(final String... args) {
        System.out.println("Initializing the driver");
        qldbDriver = QldbDriver.builder()
            .ledger("quick-start")
            .transactionRetryPolicy(RetryPolicy
                .builder()
```

```
                .maxRetries(3)
                .build())
            .sessionClientBuilder(QldbSessionClient.builder())
            .build();
    }
}
```

Schritt 3: Erstellen einer Tabelle und eines Index

Im folgenden Codebeispiel wird veranschaulicht, wie `CREATE TABLE`- und `CREATE INDEX`-Anweisungen ausgeführt werden.

Fügen Sie in der`main` Methode den folgenden Code hinzu, der einen Tabellennamen`People` und einen Index für das`lastName` Feld in dieser Tabelle erstellt. Indizes sind erforderlich, um die Abfrageleistung zu optimieren und dabei zu helfen, Konfliktausnahmen für Optimist Concurrency Control (OCC) zu begrenzen.

```
// Create a table and an index in the same transaction
qldbDriver.execute(txn -> {
    System.out.println("Creating a table and an index");
    txn.execute("CREATE TABLE People");
    txn.execute("CREATE INDEX ON People(lastName)");
});
```

Schritt 4: Einfügen eines Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine `INSERT`-Anweisung ausgeführt wird. QLDB unterstützt die PartiQL-Abfragesprache (SQL-kompatibel) und das Amazon Ion-Datenformat (Superset von JSON).

Fügen Sie den folgenden Code hinzu, der ein Dokument in die Tabelle `People` einfügt.

```
// Insert a document
qldbDriver.execute(txn -> {
    System.out.println("Inserting a document");
    IonStruct person = ionSys.newEmptyStruct();
    person.put("firstName").newString("John");
    person.put("lastName").newString("Doe");
    person.put("age").newInt(32);
    txn.execute("INSERT INTO People ?", person);
```

```
});
```

In diesem Beispiel wird ein Fragezeichen (?) als Variablenplatzhalter verwendet, um die Dokumentinformationen an die Anweisung zu übergeben. Wenn Sie Platzhalter verwenden, müssen Sie einen Wert vom Typ `IonValue` übergeben.

> **Tip**
>
> Um mehrere Dokumente mithilfe einer einzigenINSERT Anweisung einzufügen, können Sie der Anweisung wie folgt einen Parameter vom Typ IonList(explizit als`IonValue`) übergeben.
>
> ```
> // people is an IonList explicitly cast as an IonValue
> txn.execute("INSERT INTO People ?", (IonValue) people);
> ```
>
> Sie schließen den Variablen-Platzhalter (?) nicht in doppelte eckige Klammern (`<<...>>`) ein, wenn Sie eine übergeben`IonList`. In manuellen PartiQL-Anweisungen bezeichnen doppelte spitze Klammern eine ungeordnete Sammlung, die als Bag bezeichnet wird.

Schritt 5: Abfragen des Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine `SELECT`-Anweisung ausgeführt wird.

Fügen Sie den folgenden Code hinzu, der ein Dokument aus der Tabelle `People` abfragt.

```
// Query the document
qldbDriver.execute(txn -> {
    System.out.println("Querying the table");
    Result result = txn.execute("SELECT * FROM People WHERE lastName = ?",
            ionSys.newString("Doe"));
    IonStruct person = (IonStruct) result.iterator().next();
    System.out.println(person.get("firstName")); // prints John
    System.out.println(person.get("lastName")); // prints Doe
    System.out.println(person.get("age")); // prints 32
});
```

Schritt 6: Aktualisieren des Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine `UPDATE`-Anweisung ausgeführt wird.

1. Fügen Sie den folgenden Code hinzu, der ein Dokument in der`People` Tabelle aktualisiert, in`age` es auf aktualisiert wird`42`.

```
// Update the document
qldbDriver.execute(txn -> {
    System.out.println("Updating the document");
    final List<IonValue> parameters = new ArrayList<>();
    parameters.add(ionSys.newInt(42));
    parameters.add(ionSys.newString("Doe"));
    txn.execute("UPDATE People SET age = ? WHERE lastName = ?", parameters);
});
```

2. Fragen Sie das Dokument erneut ab, um den aktualisierten Wert zu sehen.

```
// Query the updated document
qldbDriver.execute(txn -> {
    System.out.println("Querying the table for the updated document");
    Result result = txn.execute("SELECT * FROM People WHERE lastName = ?",
            ionSys.newString("Doe"));
    IonStruct person = (IonStruct) result.iterator().next();
    System.out.println(person.get("firstName")); // prints John
    System.out.println(person.get("lastName")); // prints Doe
    System.out.println(person.get("age")); // prints 32
});
```

3. Verwenden Sie Maven oder Ihre IDE, um die`App.java` Datei zu kompilieren und auszuführen.

Ausführen der vollständigen Anwendung

Das folgende Codebeispiel ist die vollständige Version der`App.java` Anwendung. Anstatt die vorherigen Schritte einzeln auszuführen, können Sie dieses Codebeispiel auch kopieren und von Anfang bis Ende ausführen. Diese Anwendung demonstriert einige grundlegende CRUD-Operationen für den Ledger namens `quick-start`.

> **Note**
>
> Bevor Sie diesen Code ausführen, stellen Sie sicher, dass Sie noch keine aktive Tabelle mit dem Namen `People` im `quick-start`-Ledger besitzen.
> Ersetzen Sie in der ersten Zeile *`project-package`* durch den`groupId` Wert, den Sie für den Maven-Befehl verwendet habenSchritt 1: Einrichten des Projekts .

```
package project-package;

import java.util.*;
import com.amazon.ion.*;
import com.amazon.ion.system.*;
import software.amazon.awssdk.services.qldbsession.QldbSessionClient;
import software.amazon.qldb.*;

public class App {
    public static IonSystem ionSys = IonSystemBuilder.standard().build();
    public static QldbDriver qldbDriver;

    public static void main(final String... args) {
        System.out.println("Initializing the driver");
        qldbDriver = QldbDriver.builder()
            .ledger("quick-start")
            .transactionRetryPolicy(RetryPolicy
                .builder()
                .maxRetries(3)
                .build())
            .sessionClientBuilder(QldbSessionClient.builder())
            .build();

        // Create a table and an index in the same transaction
        qldbDriver.execute(txn -> {
            System.out.println("Creating a table and an index");
            txn.execute("CREATE TABLE People");
            txn.execute("CREATE INDEX ON People(lastName)");
        });

        // Insert a document
        qldbDriver.execute(txn -> {
            System.out.println("Inserting a document");
            IonStruct person = ionSys.newEmptyStruct();
            person.put("firstName").newString("John");
            person.put("lastName").newString("Doe");
            person.put("age").newInt(32);
            txn.execute("INSERT INTO People ?", person);
        });

        // Query the document
        qldbDriver.execute(txn -> {
            System.out.println("Querying the table");
```

```
            Result result = txn.execute("SELECT * FROM People WHERE lastName = ?",
                    ionSys.newString("Doe"));
            IonStruct person = (IonStruct) result.iterator().next();
            System.out.println(person.get("firstName")); // prints John
            System.out.println(person.get("lastName")); // prints Doe
            System.out.println(person.get("age")); // prints 32
        });

        // Update the document
        qldbDriver.execute(txn -> {
            System.out.println("Updating the document");
            final List<IonValue> parameters = new ArrayList<>();
            parameters.add(ionSys.newInt(42));
            parameters.add(ionSys.newString("Doe"));
            txn.execute("UPDATE People SET age = ? WHERE lastName = ?", parameters);
        });

        // Query the updated document
        qldbDriver.execute(txn -> {
            System.out.println("Querying the table for the updated document");
            Result result = txn.execute("SELECT * FROM People WHERE lastName = ?",
                    ionSys.newString("Doe"));
            IonStruct person = (IonStruct) result.iterator().next();
            System.out.println(person.get("firstName")); // prints John
            System.out.println(person.get("lastName")); // prints Doe
            System.out.println(person.get("age")); // prints 42
        });
    }
}
```

Verwenden Sie Maven oder Ihre IDE, um die`App.java` Datei zu kompilieren und auszuführen.

Amazon QLDB-Treiber für Java — Kochbuch-Referenz

Dieses Referenzhandbuch zeigt häufige Anwendungsfälle des Amazon QLDB-Treibers für Java. Es enthält Java-Codebeispiele, die zeigen, wie CRUD-Vorgänge (Erstellen, Lesen, Aktualisieren, Löschen) verwendet werden müssen. Es enthält auch Codebeispiele für die Verarbeitung von Amazon Ion-Daten. Darüber hinaus werden in diesem Leitfaden bewährte Verfahren vorgestellt, um Transaktionen idempotent zu machen und Eindeutigkeitsbeschränkungen zu implementieren.

> **Note**
>
> Gegebenenfalls haben einige Anwendungsfälle unterschiedliche Codebeispiele für jede unterstützte Hauptversion des QLDB-Treibers für Java.

Inhalt

Importieren des Treibers

Das folgende Codebeispiel importiert den Treiber, den QLDB-Sitzungsclient, Amazon Ion-Pakete und andere verwandte Abhängigkeiten.

2.x

```
import com.amazon.ion.IonStruct;
```

```
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;

import software.amazon.awssdk.services.qldbsession.QldbSessionClient;
import software.amazon.qldb.QldbDriver;
import software.amazon.qldb.Result;
```

1.x

```
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;

import com.amazonaws.services.qldbsession.AmazonQLDBSessionClientBuilder;
import software.amazon.qldb.PooledQldbDriver;
import software.amazon.qldb.Result;
```

Instanziieren des Treibers

Im folgenden Codebeispiel wird eine Treiberinstanz erstellt, die eine Verbindung zu einem angegebenen Ledger-Namen herstellt und die angegebene Wiederholungslogik mit einem benutzerdefinierten Wiederholungslimit verwendet.

> **Note**
>
> In diesem Beispiel wird auch ein Amazon Ion-Systemobjekt (`IonSystem`) instanziiert. Sie benötigen dieses Objekt, um Ionen-Daten zu verarbeiten, wenn Sie einige Datenoperationen in dieser Referenz ausführen. Weitere Informationen hierzu finden Sie unter Arbeiten mit Amazon Ion-Technologie.

2.x

```
QldbDriver qldbDriver = QldbDriver.builder()
    .ledger("vehicle-registration")
    .transactionRetryPolicy(RetryPolicy
        .builder()
        .maxRetries(3)
```

```
        .build())
    .sessionClientBuilder(QldbSessionClient.builder())
    .build();

IonSystem SYSTEM = IonSystemBuilder.standard().build();
```

1.x

```
PooledQldbDriver qldbDriver = PooledQldbDriver.builder()
    .withLedger("vehicle-registration")
    .withRetryLimit(3)
    .withSessionClientBuilder(AmazonQLDBSessionClientBuilder.standard())
    .build();

IonSystem SYSTEM = IonSystemBuilder.standard().build();
```

CRUD-Operationen

QLDB führt CRUD-Operationen (Erstellen, Lesen, Aktualisieren, Löschen) als Teil einer Transaktion durch.

> **Warning**
>
> Als bewährte Methode sollten Sie Ihre Schreibtransaktionen strikt idempotent gestalten.

Transaktionen idempotent machen

Wir empfehlen, Schreibtransaktionen idempotent zu machen, um unerwartete Nebenwirkungen bei Wiederholungsversuchen zu vermeiden. Eine Transaktion ist idempotent, wenn sie mehrfach ausgeführt werden kann und jedes Mal zu identischen Ergebnissen führt.

Stellen Sie sich zum Beispiel eine Transaktion vor, bei der ein Dokument in eine Tabelle mit dem Namen eingefügt wird`Person`. Die Transaktion sollte zunächst prüfen, ob das Dokument bereits in der Tabelle vorhanden ist. Ohne diese Überprüfung enthält die Tabelle möglicherweise doppelte Dokumente.

Nehmen wir an, QLDB führt die Transaktion auf der Serverseite erfolgreich durch, aber der Client läuft beim Warten auf eine Antwort ein Timeout. Wenn die Transaktion nicht idempotent ist, kann dasselbe Dokument im Falle eines erneuten Versuchs mehrmals eingefügt werden.

Verwendung von Indizes zur Vermeidung vollständiger Tabellenscans

Wir empfehlen außerdem, Anweisungen mit einer`WHERE` Prädikatklausel auszuführen, indem Sie einen Gleichheitsoperator für ein indiziertes Feld oder eine Dokument-ID verwenden, z. B.`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`. Ohne diese indizierte Suche muss QLDB einen Tabellenscan durchführen, was zu Transaktions-Timeouts oder OCC-Konflikten (Optimist Concurrency Control) führen kann.

Weitere Informationen zu OCC finden Sie unter Amazon QLDB-Nebenläufigkeit von Amazon QLDB-Nebenläufigkeit.

Implizit erstellte Transaktionen

Die Methode QldbDriver.execute akzeptiert eine Lambda-Funktion, die eine Instanz von Executor empfängt, mit der Sie Anweisungen ausführen können. Die`Executor` Instanz umschließt eine implizit erstellte Transaktion.

Sie können Anweisungen innerhalb der Lambda-Funktion ausführen, indem Sie die`Executor.execute` Methode verwenden. Der Treiber schreibt die Transaktion implizit fest, wenn die Lambda-Funktion zurückkehrt.

In den folgenden Abschnitten wird gezeigt, wie grundlegende CRUD-Operationen ausgeführt, eine benutzerdefinierte Wiederholungslogik angegeben und Eindeutigkeitsbeschränkungen implementiert werden.

> **Note**
>
> Gegebenenfalls enthalten diese Abschnitte Codebeispiele für die Verarbeitung von Amazon Ion-Daten mit der integrierten Ion-Bibliothek und der Jackson Ion Mapper-Bibliothek. Weitere Informationen hierzu finden Sie unter Arbeiten mit Amazon Ion-Technologie.

Inhalt

- Dokumente aktualisieren
- Dokumente löschen
- Ausführen mehrerer Kontoauszüge in einer Transaktion
- Wiederholungslogik
- Implementieren von Eindeutigkeitseinschränkungen

Erstellen von Tabellen

```
qldbDriver.execute(txn -> {
    txn.execute("CREATE TABLE Person");
});
```

Erstellen von Indizes

```
qldbDriver.execute(txn -> {
    txn.execute("CREATE INDEX ON Person(GovId)");
});
```

Dokumente lesen

```
// Assumes that Person table has documents as follows:
// { GovId: "TOYENC486FH", FirstName: "Brent" }

qldbDriver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM Person WHERE GovId = 'TOYENC486FH'");
    IonStruct person = (IonStruct) result.iterator().next();
    System.out.println(person.get("GovId")); // prints TOYENC486FH
    System.out.println(person.get("FirstName")); // prints Brent
});
```

Verwendung von Abfrageparametern

Das folgende Codebeispiel verwendet einen Abfrageparameter vom Typ Ion.

```
qldbDriver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM Person WHERE GovId = ?",
            SYSTEM.newString("TOYENC486FH"));
    IonStruct person = (IonStruct) result.iterator().next();
    System.out.println(person.get("GovId")); // prints TOYENC486FH
    System.out.println(person.get("FirstName")); // prints Brent
```

```
});
```

Das folgende Codebeispiel verwendet mehrere Abfrageparameter.

```
qldbDriver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM Person WHERE GovId = ? AND FirstName
 = ?",
            SYSTEM.newString("TOYENC486FH"),
            SYSTEM.newString("Brent"));
    IonStruct person = (IonStruct) result.iterator().next();
    System.out.println(person.get("GovId")); // prints TOYENC486FH
    System.out.println(person.get("FirstName")); // prints Brent
});
```

Das folgende Codebeispiel verwendet eine Liste von Abfrageparametern.

```
qldbDriver.execute(txn -> {
    final List<IonValue> parameters = new ArrayList<>();
    parameters.add(SYSTEM.newString("TOYENC486FH"));
    parameters.add(SYSTEM.newString("ROEE1"));
    parameters.add(SYSTEM.newString("YH844"));
    Result result = txn.execute("SELECT * FROM Person WHERE GovId IN (?,?,?)",
 parameters);
    IonStruct person = (IonStruct) result.iterator().next();
    System.out.println(person.get("GovId")); // prints TOYENC486FH
    System.out.println(person.get("FirstName")); // prints Brent
});
```

Den Jackson-Mapper verwenden

```
// Assumes that Person table has documents as follows:
// {GovId: "TOYENC486FH", FirstName: "Brent" }

qldbDriver.execute(txn -> {
    try {
        Result result = txn.execute("SELECT * FROM Person WHERE GovId =
 'TOYENC486FH'");
        Person person = MAPPER.readValue(result.iterator().next(), Person.class);
        System.out.println(person.getFirstName()); // prints Brent
        System.out.println(person.getGovId()); // prints TOYENC486FH
    } catch (IOException e) {
        e.printStackTrace();
```

```
    }
});
```

Verwendung von Abfrageparametern

Das folgende Codebeispiel verwendet einen Abfrageparameter vom Typ Ion.

```
qldbDriver.execute(txn -> {
    try {
        Result result = txn.execute("SELECT * FROM Person WHERE GovId = ?",
                MAPPER.writeValueAsIonValue("TOYENC486FH"));
        Person person = MAPPER.readValue(result.iterator().next(), Person.class);
        System.out.println(person.getFirstName()); // prints Brent
        System.out.println(person.getGovId()); // prints TOYENC486FH
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

Das folgende Codebeispiel verwendet mehrere Abfrageparameter.

```
qldbDriver.execute(txn -> {
    try {
        Result result = txn.execute("SELECT * FROM Person WHERE GovId = ? AND FirstName
 = ?",
                MAPPER.writeValueAsIonValue("TOYENC486FH"),
                MAPPER.writeValueAsIonValue("Brent"));
        Person person = MAPPER.readValue(result.iterator().next(), Person.class);
        System.out.println(person.getFirstName()); // prints Brent
        System.out.println(person.getGovId()); // prints TOYENC486FH
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

Das folgende Codebeispiel verwendet eine Liste von Abfrageparametern.

```
qldbDriver.execute(txn -> {
    try {
        final List<IonValue> parameters = new ArrayList<>();
        parameters.add(MAPPER.writeValueAsIonValue("TOYENC486FH"));
        parameters.add(MAPPER.writeValueAsIonValue("ROEE1"));
```

```
        parameters.add(MAPPER.writeValueAsIonValue("YH844"));
        Result result = txn.execute("SELECT * FROM Person WHERE GovId IN (?,?,?)",
 parameters);
        Person person = MAPPER.readValue(result.iterator().next(), Person.class);
        System.out.println(person.getFirstName()); // prints Brent
        System.out.println(person.getGovId()); // prints TOYENC486FH
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

> **Note**
>
> Wenn Sie eine Abfrage ohne eine indizierte Suche ausführen, wird ein vollständiger Tabellenscan aufgerufen. In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Abfragen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Einfügen von Dokumenten

Das folgende Codebeispiel fügt Ion-Datentypen ein.

```
qldbDriver.execute(txn -> {
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    Result result = txn.execute("SELECT * FROM Person WHERE GovId = ?",
            SYSTEM.newString("TOYENC486FH"));
    // Check if there is a result
    if (!result.iterator().hasNext()) {
        IonStruct person = SYSTEM.newEmptyStruct();
        person.put("GovId").newString("TOYENC486FH");
        person.put("FirstName").newString("Brent");
        // Insert the document
        txn.execute("INSERT INTO Person ?", person);
    }
});
```

Den Jackson-Mapper verwenden

Das folgende Codebeispiel fügt Ion-Datentypen ein.

```
qldbDriver.execute(txn -> {
    try {
        // Check if a document with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
        Result result = txn.execute("SELECT * FROM Person WHERE GovId = ?",
                MAPPER.writeValueAsIonValue("TOYENC486FH"));
        // Check if there is a result
        if (!result.iterator().hasNext()) {
            // Insert the document
            txn.execute("INSERT INTO Person ?",
                    MAPPER.writeValueAsIonValue(new Person("Brent", "TOYENC486FH")));
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

Diese Transaktion fügt ein Dokument in die`Person` Tabelle ein. Vor dem Einfügen wird zunächst geprüft, ob das Dokument bereits in der Tabelle vorhanden ist. Diese Prüfung macht die Transaktion idempotenter Natur. Selbst wenn Sie diese Transaktion mehrmals ausführen, verursacht sie keine unbeabsichtigten Nebenwirkungen.

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Einfügen mehrerer Dokumente in eine Anweisung

Um mehrere Dokumente mithilfe einer einzigenINSERT Anweisung einzufügen, können Sie der Anweisung wie folgt einen Parameter vom Typ IonList(explizit als`IonValue`) übergeben.

```
// people is an IonList explicitly cast as an IonValue
txn.execute("INSERT INTO People ?", (IonValue) people);
```

Sie schließen den variablen Platzhalter (?) nicht in doppelte eckige Klammern (`<<...>>`) ein, wenn Sie eine übergeben`IonList`. In manuellen PartiQL-Anweisungen bezeichnen doppelte spitze Klammern eine ungeordnete Sammlung, die als Bag bezeichnet wird.

Warum ist die explizite Besetzung erforderlich?

Die Methode TransactionExecutor.execute ist überlastet. Es akzeptiert eine variable Anzahl von`IonValue` Argumenten (varargs) oder ein einzelnes`List<IonValue>` Argument. In Ion-Java`IonList` ist es implementiert als`List<IonValue>`.

Java verwendet standardmäßig die spezifischste Methodenimplementierung, wenn Sie eine überladene Methode aufrufen. In diesem Fall wird, wenn Sie einen`IonList` Parameter übergeben, standardmäßig die Methode verwendet, die a verwendet`List<IonValue>`. Beim Aufruf übergibt diese Methodenimplementierung die`IonValue` Elemente der Liste als unterschiedliche Werte. Um stattdessen die Methode varargs aufzurufen, müssen Sie einen`IonList` Parameter explizit in einen umwandeln`IonValue`.

Dokumente aktualisieren

```
qldbDriver.execute(txn -> {
    final List<IonValue> parameters = new ArrayList<>();
    parameters.add(SYSTEM.newString("John"));
    parameters.add(SYSTEM.newString("TOYENC486FH"));
    txn.execute("UPDATE Person SET FirstName = ? WHERE GovId = ?", parameters);
});
```

Den Jackson-Mapper verwenden

```
qldbDriver.execute(txn -> {
    try {
        final List<IonValue> parameters = new ArrayList<>();
        parameters.add(MAPPER.writeValueAsIonValue("John"));
        parameters.add(MAPPER.writeValueAsIonValue("TOYENC486FH"));
        txn.execute("UPDATE Person SET FirstName = ? WHERE GovId = ?", parameters);
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktivierten`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Dokumente löschen

```
qldbDriver.execute(txn -> {
    txn.execute("DELETE FROM Person WHERE GovId = ?",
            SYSTEM.newString("TOYENC486FH"));
});
```

Den Jackson-Mapper verwenden

```
qldbDriver.execute(txn -> {
    try {
        txn.execute("DELETE FROM Person WHERE GovId = ?",
                MAPPER.writeValueAsIonValue("TOYENC486FH"));
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Ausführen mehrerer Kontoauszüge in einer Transaktion

```
// This code snippet is intentionally trivial. In reality you wouldn't do this because
 you'd
// set your UPDATE to filter on vin and insured, and check if you updated something or
 not.
public static boolean InsureCar(QldbDriver qldbDriver, final String vin) {
    final IonSystem ionSystem = IonSystemBuilder.standard().build();
    final IonString ionVin = ionSystem.newString(vin);

    return qldbDriver.execute(txn -> {
        Result result = txn.execute(
                "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
                ionVin);
        if (!result.isEmpty()) {
            txn.execute("UPDATE Vehicles SET insured = TRUE WHERE vin = ?", ionVin);
```

```
            return true;
        }
        return false;
    });
}
```

Wiederholungslogik

Die`execute` Methode des Treibers verfügt über einen integrierten Wiederholungsmechanismus, der die Transaktion erneut versucht, wenn eine wiederholbare Ausnahme auftritt (z. B. Timeouts oder OCC-Konflikte).

2.x

Die maximale Anzahl der Wiederholungsversuche und die Backoff-Strategie sind konfigurierbar.

Das Standardlimit für Wiederholungsversuche ist4 und die Standard-Backoff-Strategie ist DefaultQldbTransactionBackoffStrategy. Sie können die Wiederholungskonfiguration pro Treiberinstanz und auch pro Transaktion festlegen, indem Sie eine Instanz von verwenden RetryPolicy.

Das folgende Codebeispiel spezifiziert die Wiederholungslogik mit einem benutzerdefinierten Wiederholungslimit und einer benutzerdefinierten Backoff-Strategie für eine Instanz des Treibers.

```
public void retry() {
    QldbDriver qldbDriver = QldbDriver.builder()
            .ledger("vehicle-registration")
            .transactionRetryPolicy(RetryPolicy.builder()
                    .maxRetries(2)
                    .backoffStrategy(new CustomBackOffStrategy()).build())
            .sessionClientBuilder(QldbSessionClient.builder())
            .build();
}

private class CustomBackOffStrategy implements BackoffStrategy {

    @Override
    public Duration calculateDelay(RetryPolicyContext retryPolicyContext) {
        return Duration.ofMillis(1000 * retryPolicyContext.retriesAttempted());
    }
}
```

Das folgende Codebeispiel spezifiziert die Wiederholungslogik mit einem benutzerdefinierten Wiederholungslimit und einer benutzerdefinierten Backoff-Strategie für eine bestimmte Transaktion. Diese Konfiguration für`execute` überschreibt die Wiederholungslogik, die für die Treiberinstanz festgelegt ist.

```
public void retry() {
    Result result = qldbDriver.execute(txn -> { txn.execute("SELECT * FROM Person
 WHERE GovId = ?",
            SYSTEM.newString("TOYENC486FH")); },
            RetryPolicy.builder()
                    .maxRetries(2)
                    .backoffStrategy(new CustomBackOffStrategy())
                    .build());
}

private class CustomBackOffStrategy implements BackoffStrategy {

    // Configuring a custom backoff which increases delay by 1s for each attempt.
    @Override
    public Duration calculateDelay(RetryPolicyContext retryPolicyContext) {
        return Duration.ofMillis(1000 * retryPolicyContext.retriesAttempted());
    }
}
```

1.x

Die maximale Anzahl der Wiederholungsversuche ist konfigurierbar. Sie können das Wiederholungslimit konfigurieren, indem Sie die`retryLimit` Eigenschaft bei der Initialisierung festlegen`PooledQldbDriver`.

Das Standardlimit für Wiederholungsversuche ist`4`.

Implementieren von Eindeutigkeitseinschränkungen

QLDB unterstützt keine eindeutigen Indizes, aber Sie können dieses Verhalten in Ihrer Anwendung implementieren.

Angenommen, Sie möchten eine Eindeutigkeitsbeschränkung für das`GovId` Feld in der`Person` Tabelle implementieren. Dazu können Sie eine Transaktion schreiben, die Folgendes bewirkt:

1. Stellen Sie sicher, dass die Tabelle keine vorhandenen Dokumente mit einem bestimmten Wert enthält`GovId`.
2. Fügen Sie das Dokument ein, wenn die Assertion erfolgreich ist.

Besteht eine konkurrierende Transaktion gleichzeitig die Assertion, wird nur eine der Transaktionen erfolgreich abgeschlossen. Die andere Transaktion schlägt mit einer OCC-Konfliktexception fehl.

Das folgende Codebeispiel veranschaulicht, wie diese Einheitseinschränungslogik implementiert werden muss.

```
qldbDriver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM Person WHERE GovId = ?",
            SYSTEM.newString("TOYENC486FH"));
    // Check if there is a result
    if (!result.iterator().hasNext()) {
        IonStruct person = SYSTEM.newEmptyStruct();
        person.put("GovId").newString("TOYENC486FH");
        person.put("FirstName").newString("Brent");
        // Insert the document
        txn.execute("INSERT INTO Person ?", person);
    }
});
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Arbeiten mit Amazon Ion-Technologie

Es gibt mehrere Möglichkeiten, Amazon-Ion-Daten in QLDB zu verarbeiten. Sie können integrierte Methoden aus der Ion-Bibliothek verwenden, um Dokumente nach Bedarf flexibel zu erstellen und zu ändern. Oder Sie können das Jackson-Datenformatmodul für Ion von FasterXML verwenden, um Ion-Dokumente einfachen alten Java-Objektmodellen (POJO) zuzuordnen.

Die folgenden Abschnitte enthalten Codebeispiele für die Verarbeitung von Ion-Daten mit beiden Techniken.

Inhalt

Import der Ion-Pakete

Fügen Sie das Artefakt ion-java als Abhängigkeit zu Ihrem Java-Projekt hinzu.

Gradle

```
dependencies {
    compile group: 'com.amazon.ion', name: 'ion-java', version: '1.6.1'
}
```

Maven

```
<dependencies>
  <dependency>
    <groupId>com.amazon.ion</groupId>
    <artifactId>ion-java</artifactId>
    <version>1.6.1</version>
  </dependency>
</dependencies>
```

Importieren Sie die folgenden Ion-Pakete.

```
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonSystem;
import com.amazon.ion.system.IonSystemBuilder;
```

Den Jackson-Mapper verwenden

Fügen Sie das Artefakt jackson-dataformat-ionals Abhängigkeit zu Ihrem Java-Projekt hinzu. QLDB erfordert Version`2.10.0` oder höher.

Gradle

```
dependencies {
    compile group: 'com.fasterxml.jackson.dataformat', name: 'jackson-dataformat-
ion', version: '2.10.0'
}
```

Maven

```
<dependencies>
  <dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-ion</artifactId>
    <version>2.10.0</version>
  </dependency>
</dependencies>
```

Importieren Sie die folgenden Ion-Pakete.

```
import com.amazon.ion.IonReader;
import com.amazon.ion.IonStruct;
import com.amazon.ion.system.IonReaderBuilder;
import com.amazon.ion.system.IonSystemBuilder;

import com.fasterxml.jackson.dataformat.ion.IonObjectMapper;
import com.fasterxml.jackson.dataformat.ion.ionvalue.IonValueMapper;
```

Initialisieren/Ion-Elemente

```
IonSystem SYSTEM = IonSystemBuilder.standard().build();
```

Den Jackson-Mapper verwenden

```
IonObjectMapper MAPPER = new IonValueMapper(IonSystemBuilder.standard().build());
```

Erstellen von Ion-Objekten

Das folgende Codebeispiel erstellt ein Ion-Objekt mithilfe der`IonStruct` Schnittstelle und ihrer integrierten Methoden.

```
IonStruct ionStruct = SYSTEM.newEmptyStruct();

ionStruct.put("GovId").newString("TOYENC486FH");
ionStruct.put("FirstName").newString("Brent");

System.out.println(ionStruct.toPrettyString()); // prints a nicely formatted copy of
 ionStruct
```

Den Jackson-Mapper verwenden

Angenommen, Sie haben eine JSON-zugeordnete Modellklasse wie folgt benannt`Person`.

```
import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;

public static class Person {
    private final String firstName;
    private final String govId;

    @JsonCreator
    public Person(@JsonProperty("FirstName") final String firstName,
                  @JsonProperty("GovId") final String govId) {
        this.firstName = firstName;
        this.govId = govId;
    }

    @JsonProperty("FirstName")
    public String getFirstName() {
        return firstName;
    }

    @JsonProperty("GovId")
    public String getGovId() {
        return govId;
    }
}
```

Das folgende Codebeispiel erstellt ein`IonStruct` Objekt aus einer Instanz von`Person`.

```
IonStruct ionStruct = (IonStruct) MAPPER.writeValueAsIonValue(new Person("Brent",
 "TOYENC486FH"));
```

Lesen von Ion-Objekten

Das folgende Codebeispiel druckt jedes Feld der`ionStruct` Instanz.

```
// ionStruct is an instance of IonStruct
System.out.println(ionStruct.get("GovId")); // prints TOYENC486FH
System.out.println(ionStruct.get("FirstName")); // prints Brent
```

Den Jackson-Mapper verwenden

Das folgende Codebeispiel liest ein`IonStruct` Objekt und ordnet es einer Instanz von zu`Person`.

```
// ionStruct is an instance of IonStruct
IonReader reader = IonReaderBuilder.standard().build(ionStruct);
Person person = MAPPER.readValue(reader, Person.class);
System.out.println(person.getFirstName()); // prints Brent
System.out.println(person.getGovId()); // prints TOYENC486FH
```

Weitere Informationen zur Arbeit mit Ion finden Sie in der Amazon Ion-Dokumentation unter GitHub. Weitere Codebeispiele für die Arbeit mit Ion in QLDB finden Sie unterArbeiten mit Amazon Ion-Datentypen in Amazon QLDB.

Amazon QLDB-Treiber für.NET

Um mit Daten in Ihrem Ledger zu arbeiten, können Sie von Ihrer Microsoft.NET-Anwendung aus mithilfe einesAWS bereitgestellten Treibers eine Verbindung zu Amazon QLDB herstellen. Der Treiber zielt auf .NET Standard 2.0 ab. Insbesondere unterstützt es .NET Core (LTS) 2.1+ und .NET Framework 4.5.2+. Informationen zur Kompatibilität finden Sie unter.NET Standard auf der Microsoft Docs-Website.

Wir empfehlen dringend, den Ion-Objektmapper zu verwenden, um die Notwendigkeit einer manuellen Konvertierung zwischen Amazon Ion-Typen und nativen C#-Typen vollständig zu umgehen.

Die folgenden Themen beschreiben die ersten Schritte mit dem QLDB-Treiber für.NET.

Themen

- Ressourcen für Fahrer

- Voraussetzungen
- Installation
- Amazon QLDB-Treiber für.NET — Schnellstart-Tutorial
- Amazon QLDB-Treiber für.NET — Kochbuch-Referenz

Ressourcen für Fahrer

Weitere Informationen zu den vom .NET-Treiber unterstützten Funktionen finden Sie in den folgenden Ressourcen:

- API-Referenz
- Treiber-Quellcode (GitHub)
- Quellcode der Beispielanwendung (GitHub)
- Amazon Ion Cookbook
- Ion Objektmapper (GitHub)

Voraussetzungen

Bevor Sie mit dem QLDB-Treiber für.NET beginnen, müssen Sie die folgenden Schritte durchführen:

1. Befolgen Sie die AWS-Einrichtungsanweisungen unter Zugreifen auf Amazon QLDB. Diese umfasst die folgenden Funktionen:
    1. Registrieren beiAWS.
    2. Erstellen Sie einen Benutzer mit den entsprechenden QLDB-Berechtigungen.
    3. Erteilen programmgesteuerten Zugriffs bei der Entwicklung.
2. Laden Sie das .NET Core SDK Version 2.1 oder höher von der Microsoft-Website mit .NET-Downloads herunter und installieren Sie es.
3. (Optional) Installieren Sie eine Integrated Development Environment (IDE, integrierte Entwicklungsumgebung) Ihrer Wahl, z. B. Visual Studio, Visual Studio für Mac oder Visual Studio Code. Sie können diese von der Microsoft Visual Studio-Website herunterladen.
4. Konfigurieren Sie die Entwicklungsumgebung für das AWS SDK for .NET:
    1. Richten Sie Ihre AWS-Anmeldeinformationen ein. Wir empfehlen die Erstellung einer freigegebenen Anmeldeinformationsdatei.

Anweisungen finden Sie im AWS SDK for .NETEntwicklerhandbuch unter Konfiguration vonAWS Anmeldeinformationen mithilfe einer Anmeldeinformationsdatei.

2. Stellen Sie Ihre Standardeinstellung einAWS-Region. Weitere Informationen erhalten Sie AWS-Regionunter.

   Eine vollständige Liste der verfügbaren Regionen finden Sie unter Amazon QLDB-Endpunkte und Kontingente in der Allgemeine AWS-Referenz.

Als Nächstes können Sie eine einfache Beispielanwendung einrichten und kurze Codebeispiele ausführen — oder Sie können den Treiber in einem vorhandenen .NET-Projekt installieren.

- Um den QLDB-Treiber und denAWS SDK for .NET in einem vorhandenen Projekt zu installieren, fahren Sie mit fortInstallation.
- Informationen zum Einrichten eines Projekts und zur Ausführung von kurzen Codebeispielen, die grundlegende Datentransaktionen in einem Ledger veranschaulichen, finden Sie unterErste-Schritt.

Installation

Verwenden Sie den NuGet Paketmanager, um den QLDB-Treiber für.NET zu installieren. Wir empfehlen, Visual Studio oder eine IDE Ihrer Wahl zu verwenden, um Projektabhängigkeiten hinzuzufügen. Der Name des Treiberpakets ist Amazon.QLDB.Driver.

Öffnen Sie beispielsweise in Visual Studio die NuGet Package Manager Console im Menü Tools. Geben Sie anschließend den folgenden Befehl an der `PM>`-Eingabeaufforderung ein:

```
PM> Install-Package Amazon.QLDB.Driver
```

Durch die Installation des Treibers werden auch seine Abhängigkeiten installiert, einschließlich der PaketeAWS SDK for .NET und Amazon Ion.

Installieren Sie den Ion Object Mapper

Version 1.3.0 des QLDB-Treibers für.NET bietet Unterstützung für die Annahme und Rückgabe nativer C#-Datentypen, ohne dass Sie mit Amazon Ion arbeiten müssen. Um diese Funktion zu verwenden, fügen Sie Ihrem Projekt das folgende Paket hinzu.

- Amazon.Qldb.Driver.Serialization — Eine Bibliothek, die Ion-Werte einfachen alten C#-CLR-Objekten (POCO) zuordnen kann und umgekehrt. Mit diesem Ion-Objektmapper kann Ihre Anwendung direkt mit nativen C#-Datentypen interagieren, ohne mit Ion arbeiten zu müssen. Eine kurze Anleitung zur Verwendung dieser Bibliothek finden Sie in der Datei Serialization.md im GitHub Repository`awslabs/amazon-qldb-driver-dotnet`.

Geben Sie den folgenden Befehl ein, um dieses Paket zu installieren.

```
PM> Install-Package Amazon.QLDB.Driver.Serialization
```

Kurze Codebeispiele für die Ausführung grundlegender Datentransaktionen in einem Ledger finden Sie unterKochbuchreferenz.

Amazon QLDB-Treiber für.NET — Schnellstart-Tutorial

In diesem Tutorial erfahren Sie, wie Sie eine einfache Anwendung mit dem Amazon-QLDB-T Dieses Handbuch enthält Schritte zum Installieren des Treibers und kurze Codebeispiele für grundlegende CRUD-Vorgänge (Create, Read, Update und Delete).

Themen

Voraussetzungen

Bevor Sie beginnen, stellen Sie sicher, dass die folgende Voraussetzung erfüllt ist:

1. Führen Sie die Voraussetzungen für den .NET-Treiber aus, sofern Sie dies noch nicht getan haben. Dazu gehören die RegistrierungAWS, die Gewährung von programmatischem Zugriff für die Entwicklung und die Installation des.NET Core SDK.

2. Ledger mit dem Namen `quick-start` erstellen.

   Informationen zum Erstellen eines Ledgers finden Sie unterGrundfunktionen für Amazon QLDB-Ledgers oderSchritt 1: Erstellen eines neuen Ledgers unter Erste Schritte mit der Konsole.

Schritt 1: Einrichten des Projekts

Richten Sie zuerst Ihr .NET-Projekt ein.

1. Um eine Vorlagenanwendung zu erstellen und auszuführen, geben Sie die folgendendotnet Befehle auf einem Terminal ein PowerShell, z. B. bash oder Command Prompt.

   ```
   $ dotnet new console --output Amazon.QLDB.QuickStartGuide
   $ dotnet run --project Amazon.QLDB.QuickStartGuide
   ```

   Diese Vorlage erstellt einen Ordner mit dem Namen `Amazon.QLDB.QuickStartGuide`. In diesem Ordner wird ein Projekt mit dem gleichen Namen und einer Datei mit dem Namen `Program.cs` erstellt. Das Programm enthält Code, der die Ausgabe `Hello World!` anzeigt.
2. Verwenden Sie den NuGet Paketmanager, um den QLDB-Treiber für.NET zu installieren. Wir empfehlen, Visual Studio oder eine IDE Ihrer Wahl zu verwenden, um Ihrem Projekt Abhängigkeiten hinzuzufügen. Der Name des Treiberpakets ist Amazon.QLDB.Driver.
   - Öffnen Sie beispielsweise in Visual Studio die NuGet Package Manager Console im Menü Tools. Geben Sie anschließend den folgenden Befehl an der `PM>`-Eingabeaufforderung ein:

     ```
     PM> Install-Package Amazon.QLDB.Driver
     ```
   - Oder Sie können die folgenden Befehle auf Ihrem Terminal eingeben.

     ```
     $ cd Amazon.QLDB.QuickStartGuide
     $ dotnet add package Amazon.QLDB.Driver
     ```

   Beim Installieren des Treibers werden auch die Abhängigkeiten installiert, einschließlich der AWS SDK for .NET- und der Amazon Ion-Bibliotheken.
3. Installieren Sie die Serialisierungsbibliothek des Treibers.

   ```
   PM> Install-Package Amazon.QLDB.Driver.Serialization
   ```
4. Öffnen Sie die `Program.cs` Datei.

Fügen Sie dann schrittweise die Codebeispiele in den folgenden Schritten hinzu, um einige grundlegende CRUD-Operationen auszuprobieren. Oder Sie können das step-by-step Tutorial überspringen und stattdessen die komplette Anwendung ausführen.

> **Note**
>
> - Wahl zwischen synchronen und asynchronen APIs — Der Treiber stellt synchrone und asynchrone APIs bereit. Für stark beanspruchte Anwendungen, die mehrere Anfragen verarbeiten, ohne sie zu blockieren, empfehlen wir die Verwendung der asynchronen APIs, um die Leistung zu verbessern. Der Treiber bietet synchrone APIs als zusätzlichen Komfort für bestehende Codebasen, die synchron geschrieben wurden.
>
>   Dieses Tutorial enthält sowohl synchrone als auch asynchrone Codebeispiele. Weitere Informationen zu den APIs finden Sie in den IQldbDriver - und AsyncQldbDriverI-Schnittstellen in der API-Dokumentation.
> - Verarbeitung von Amazon Ion-Daten — Dieses Tutorial enthält Codebeispiele für die standardmäßige Verarbeitung von Amazon Ion-Daten mit dem Ion-Objektmapper. QLDB führte den Ion Object Mapper in Version 1.3.0 des .NET-Treibers ein. Gegebenenfalls enthält dieses Tutorial auch Codebeispiele, bei denen die Standard-Ion-Bibliothek als Alternative verwendet wird. Weitere Informationen hierzu finden Sie unter Arbeiten mit Amazon Ion-Funktion.

Schritt 2: Initialisieren des Treibers

Initialisieren Sie eine Instance des Treibers, der eine Verbindung mit dem Ledger `quick-start` herstellt. Fügen Sie nun folgenden Code in die Datei `Program.cs` ein:

Async

```
using Amazon.QLDB.Driver;
using Amazon.QLDB.Driver.Generic;
using Amazon.QLDB.Driver.Serialization;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
```

```
    {
        public class Person
        {
            public string FirstName { get; set; }

            public string LastName { get; set; }

            public int Age { get; set; }

            public override string ToString()
            {
                return FirstName + ", " + LastName + ", " + Age.ToString();
            }
        }

        static async Task Main(string[] args)
        {
            Console.WriteLine("Create the async QLDB driver");
            IAsyncQldbDriver driver = AsyncQldbDriver.Builder()
                .WithLedger("quick-start")
                .WithSerializer(new ObjectSerializer())
                .Build();
        }
    }
}
```

Sync

```
using Amazon.QLDB.Driver;
using Amazon.QLDB.Driver.Generic;
using Amazon.QLDB.Driver.Serialization;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
    {
        public class Person
        {
            public string FirstName { get; set; }

            public string LastName { get; set; }

            public int Age { get; set; }
```

```
            public override string ToString()
            {
                return FirstName + ", " + LastName + ", " + Age.ToString();
            }
        }

        static void Main(string[] args)
        {
            Console.WriteLine("Create the sync QLDB driver");
            IQldbDriver driver = QldbDriver.Builder()
                .WithLedger("quick-start")
                .WithSerializer(new ObjectSerializer())
                .Build();
        }
    }
}
```

Verwendung der Ion-Bibliothek

Async

```
using System;
using System.Threading.Tasks;
using Amazon.IonDotnet.Tree;
using Amazon.IonDotnet.Tree.Impl;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
    {
        static IValueFactory valueFactory = new ValueFactory();

        static async Task Main(string[] args)
        {
            Console.WriteLine("Create the async QLDB driver");
            IAsyncQldbDriver driver = AsyncQldbDriver.Builder()
                .WithLedger("quick-start")
                .Build();
        }
```

```
    }
}
```

Sync

```
using System;
using Amazon.IonDotnet.Tree;
using Amazon.IonDotnet.Tree.Impl;
using Amazon.QLDB.Driver;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
    {
        static IValueFactory valueFactory = new ValueFactory();

        static void Main(string[] args)
        {
            Console.WriteLine("Create the sync QLDB Driver");
            IQldbDriver driver = QldbDriver.Builder()
                .WithLedger("quick-start")
                .Build();
        }
    }
}
```

Schritt 3: Erstellen einer Tabelle und eines Index

Für den Rest dieses Tutorials bis Schritt 6 müssen Sie die folgenden Codebeispiele an das vorherige Codebeispiel anhängen.

In diesem Schritt zeigt der folgende Code, wie`CREATE INDEX` AND-Anweisungen ausgeführt`CREATE TABLE` werden. Es erstellt eine benannte Tabelle`Person` und einen Index für das`firstName` Feld in dieser Tabelle. Indizes sind erforderlich, um die Abfrageleistung zu optimieren und dabei zu helfen, Konfliktausnahmen für Optimist Concurrency Control (OCC) zu begrenzen.

Async

```
Console.WriteLine("Creating the table and index");

// Creates the table and the index in the same transaction.
```

```
// Note: Any code within the lambda can potentially execute multiple times due to
 retries.
// For more information, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-retry-policy
await driver.Execute(async txn =>
{
    await txn.Execute("CREATE TABLE Person");
    await txn.Execute("CREATE INDEX ON Person(firstName)");
});
```

Sync

```
Console.WriteLine("Creating the tables and index");

// Creates the table and the index in the same transaction.
// Note: Any code within the lambda can potentially execute multiple times due to
 retries.
// For more information, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-retry-policy
driver.Execute(txn =>
{
    txn.Execute("CREATE TABLE Person");
    txn.Execute("CREATE INDEX ON Person(firstName)");
});
```

Schritt 4: Einfügen eines Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine `INSERT`-Anweisung ausgeführt wird. QLDB unterstützt die PartiQL-Abfragesprache (SQL-kompatibel) und das Amazon Ion-Datenformat (Superset von JSON).

Fügen Sie den folgenden Code hinzu, der ein Dokument in die Tabelle `Person` einfügt.

Async

```
Console.WriteLine("Inserting a document");

Person myPerson = new Person {
    FirstName = "John",
    LastName = "Doe",
    Age = 32
```

```
};

await driver.Execute(async txn =>
{
    IQuery<Person> myQuery = txn.Query<Person>("INSERT INTO Person ?", myPerson);
    await txn.Execute(myQuery);
});
```

Sync

```
Console.WriteLine("Inserting a document");

Person myPerson = new Person {
    FirstName = "John",
    LastName = "Doe",
    Age = 32
};

driver.Execute(txn =>
{
    IQuery<Person> myQuery = txn.Query<Person>("INSERT INTO Person ?", myPerson);
    txn.Execute(myQuery);
});
```

Verwendung der Ion-Bibliothek

Async

```
Console.WriteLine("Inserting a document");

// This is one way of creating Ion values. We can also use an IonLoader.
// For more details, see: https://docs.aws.amazon.com/qldb/latest/developerguide/
driver-cookbook-dotnet.html#cookbook-dotnet.ion
IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("firstName", valueFactory.NewString("John"));
ionPerson.SetField("lastName", valueFactory.NewString("Doe"));
ionPerson.SetField("age", valueFactory.NewInt(32));

await driver.Execute(async txn =>
{
    await txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

Sync

```
Console.WriteLine("Inserting a document");

// This is one way of creating Ion values, we can also use an IonLoader.
// For more details, see: https://docs.aws.amazon.com/qldb/latest/developerguide/
driver-cookbook-dotnet.html#cookbook-dotnet.ion
IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("firstName", valueFactory.NewString("John"));
ionPerson.SetField("lastName", valueFactory.NewString("Doe"));
ionPerson.SetField("age", valueFactory.NewInt(32));

driver.Execute(txn =>
{
    txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

Tip

Um mehrere Dokumente mithilfe einer einzigenINSERT Anweisung einzufügen, können Sie wie folgt einen Ion-Listtypparameter an die Anweisung übergeben.

```
// people is an Ion list
txn.Execute("INSERT INTO Person ?", people);
```

Sie schließen den variablen Platzhalter (?) nicht in doppelte eckige Klammern (`<<...>>`) ein, wenn Sie eine Ion-Liste übergeben. In manuellen PartiQL-Anweisungen bezeichnen doppelte spitze Klammern eine ungeordnete Sammlung, die als Bag bezeichnet wird.

Schritt 5: Abfragen des Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine `SELECT`-Anweisung ausgeführt wird.

Fügen Sie den folgenden Code hinzu, der ein Dokument aus der Tabelle `Person` abfragt.

Async

```
Console.WriteLine("Querying the table");
```

```
// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IAsyncResult<Person> selectResult = await driver.Execute(async txn =>
{
    IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person WHERE FirstName
 = ?", "John");
    return await txn.Execute(myQuery);
});

await foreach (Person person in selectResult)
{
    Console.WriteLine(person);
    // John, Doe, 32
}
```

Sync

```
Console.WriteLine("Querying the table");

// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IResult<Person> selectResult = driver.Execute(txn =>
{
    IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person WHERE FirstName
 = ?", "John");
    return txn.Execute(myQuery);
});

foreach (Person person in selectResult)
{
    Console.WriteLine(person);
    // John, Doe, 32
}
```

Verwendung der Ion-Bibliothek

Async

```
Console.WriteLine("Querying the table");

IIonValue ionFirstName = valueFactory.NewString("John");
```

```
// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IAsyncResult selectResult = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE firstName = ?",
 ionFirstName);
});

await foreach (IIonValue row in selectResult)
{
    Console.WriteLine(row.GetField("firstName").StringValue);
    Console.WriteLine(row.GetField("lastName").StringValue);
    Console.WriteLine(row.GetField("age").IntValue);
}
```

Sync

```
Console.WriteLine("Querying the table");

IIonValue ionFirstName = valueFactory.NewString("John");

// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IResult selectResult = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE firstName = ?", ionFirstName);
});

foreach (IIonValue row in selectResult)
{
    Console.WriteLine(row.GetField("firstName").StringValue);
    Console.WriteLine(row.GetField("lastName").StringValue);
    Console.WriteLine(row.GetField("age").IntValue);
}
```

In diesem Beispiel wird ein Fragezeichen (?) als Variablenplatzhalter verwendet, um die Dokumentinformationen an die Anweisung zu übergeben. Wenn Sie Platzhalter verwenden, müssen Sie einen Wert vom Typ `IonValue` übergeben.

Schritt 6: Aktualisieren des Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine `UPDATE`-Anweisung ausgeführt wird.

1. Fügen Sie den folgenden Code hinzu, der ein Dokument in der`Person` Tabelle aktualisiert, in`age` es auf 42 aktualisiert wird.

   Async

   ```
   Console.WriteLine("Updating the document");

   await driver.Execute(async txn =>
   {
       IQuery<Person> myQuery = txn.Query<Person>("UPDATE Person SET Age = ? WHERE
    FirstName = ?", 42, "John");
       await txn.Execute(myQuery);
   });
   ```

   Sync

   ```
   Console.WriteLine("Updating the document");

   driver.Execute(txn =>
   {
       IQuery<Person> myQuery = txn.Query<Person>("UPDATE Person SET Age = ? WHERE
    FirstName = ?", 42, "John");
       txn.Execute(myQuery);
   });
   ```

2. Fragen Sie das Dokument erneut ab, um den aktualisierten Wert zu sehen.

   Async

   ```
   Console.WriteLine("Querying the table for the updated document");

   IAsyncResult<Person> updateResult = await driver.Execute(async txn =>
   {
       IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person WHERE
    FirstName = ?", "John");
       return await txn.Execute(myQuery);
   });

   await foreach (Person person in updateResult)
   {
       Console.WriteLine(person);
       // John, Doe, 42
   ```

```
}
```

Sync

```
Console.WriteLine("Querying the table for the updated document");

IResult<Person> updateResult = driver.Execute(txn =>
{
    IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person WHERE
 FirstName = ?", "John");
    return txn.Execute(myQuery);
});

foreach (Person person in updateResult)
{
    Console.WriteLine(person);
    // John, Doe, 42
}
```

3. Um die Anwendung auszuführen, geben Sie den folgenden Befehl aus dem übergeordneten Verzeichnis Ihres `Amazon.QLDB.QuickStartGuide`-Projektverzeichnisses ein.

```
$ dotnet run --project Amazon.QLDB.QuickStartGuide
```

Verwendung der Ion-Bibliothek

1. Fügen Sie den folgenden Code hinzu, der ein Dokument in der`Person` Tabelle aktualisiert, in`age` es auf 42 aktualisiert wird.

   Async

```
Console.WriteLine("Updating the document");

IIonValue ionIntAge = valueFactory.NewInt(42);
IIonValue ionFirstName2 = valueFactory.NewString("John");

await driver.Execute(async txn =>
{
    await txn.Execute("UPDATE Person SET age = ? WHERE firstName = ?",
 ionIntAge, ionFirstName2);
```

```
});
```

Sync

```
Console.WriteLine("Updating a document");

IIonValue ionIntAge = valueFactory.NewInt(42);
IIonValue ionFirstName2 = valueFactory.NewString("John");

driver.Execute(txn =>
{
    txn.Execute("UPDATE Person SET age = ? WHERE firstName = ?", ionIntAge,
 ionFirstName2);
});
```

2. Fragen Sie das Dokument erneut ab, um den aktualisierten Wert zu sehen.

Async

```
Console.WriteLine("Querying the table for the updated document");

IIonValue ionFirstName3 = valueFactory.NewString("John");

IAsyncResult updateResult = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE firstName = ?",
 ionFirstName3 );
});

await foreach (IIonValue row in updateResult)
{
    Console.WriteLine(row.GetField("firstName").StringValue);
    Console.WriteLine(row.GetField("lastName").StringValue);
    Console.WriteLine(row.GetField("age").IntValue);
}
```

Sync

```
Console.WriteLine("Querying the table for the updated document");

IIonValue ionFirstName3 = valueFactory.NewString("John");
```

```
IResult updateResult = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE firstName = ?",
 ionFirstName3);
});

foreach (IIonValue row in updateResult)
{
    Console.WriteLine(row.GetField("firstName").StringValue);
    Console.WriteLine(row.GetField("lastName").StringValue);
    Console.WriteLine(row.GetField("age").IntValue);
}
```

3. Um die Anwendung auszuführen, geben Sie den folgenden Befehl aus dem übergeordneten Verzeichnis Ihres `Amazon.QLDB.QuickStartGuide`-Projektverzeichnisses ein.

```
$ dotnet run --project Amazon.QLDB.QuickStartGuide
```

Ausführen der vollständigen Anwendung

Das folgende Codebeispiel ist die vollständige Version der`Program.cs` Anwendung. Anstatt die vorherigen Schritte einzeln auszuführen, können Sie dieses Codebeispiel auch kopieren und von Anfang bis Ende ausführen. Diese Anwendung demonstriert einige grundlegende CRUD-Operationen für den Ledger namens `quick-start`.

> **Note**
>
> Bevor Sie diesen Code ausführen, stellen Sie sicher, dass Sie noch keine aktive Tabelle mit dem Namen `Person` im `quick-start`-Ledger besitzen.

Async

```
using Amazon.QLDB.Driver;
using Amazon.QLDB.Driver.Generic;
using Amazon.QLDB.Driver.Serialization;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
```

```
    {
        public class Person
        {
            public string FirstName { get; set; }

            public string LastName { get; set; }

            public int Age { get; set; }

            public override string ToString()
            {
                return FirstName + ", " + LastName + ", " + Age.ToString();
            }
        }

        static async Task Main(string[] args)
        {
            Console.WriteLine("Create the async QLDB driver");
            IAsyncQldbDriver driver = AsyncQldbDriver.Builder()
                .WithLedger("quick-start")
                .WithSerializer(new ObjectSerializer())
                .Build();

            Console.WriteLine("Creating the table and index");

            // Creates the table and the index in the same transaction.
            // Note: Any code within the lambda can potentially execute multiple
 times due to retries.
            // For more information, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-retry-policy
            await driver.Execute(async txn =>
            {
                await txn.Execute("CREATE TABLE Person");
                await txn.Execute("CREATE INDEX ON Person(firstName)");
            });

            Console.WriteLine("Inserting a document");

            Person myPerson = new Person {
                FirstName = "John",
                LastName = "Doe",
                Age = 32
            };
```

```
            await driver.Execute(async txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("INSERT INTO Person ?",
myPerson);
                await txn.Execute(myQuery);
            });

            Console.WriteLine("Querying the table");

            // The result from driver.Execute() is buffered into memory because once
the
            // transaction is committed, streaming the result is no longer possible.
            IAsyncResult<Person> selectResult = await driver.Execute(async txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person
WHERE FirstName = ?", "John");
                return await txn.Execute(myQuery);
            });

            await foreach (Person person in selectResult)
            {
                Console.WriteLine(person);
                // John, Doe, 32
            }

            Console.WriteLine("Updating the document");

            await driver.Execute(async txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("UPDATE Person SET Age
= ? WHERE FirstName = ?", 42, "John");
                await txn.Execute(myQuery);
            });

            Console.WriteLine("Querying the table for the updated document");

            IAsyncResult<Person> updateResult = await driver.Execute(async txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person
WHERE FirstName = ?", "John");
                return await txn.Execute(myQuery);
            });

            await foreach (Person person in updateResult)
```

```
            {
                Console.WriteLine(person);
                // John, Doe, 42
            }
        }
    }
}
```

Sync

```
using Amazon.QLDB.Driver;
using Amazon.QLDB.Driver.Generic;
using Amazon.QLDB.Driver.Serialization;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
    {
        public class Person
        {
            public string FirstName { get; set; }

            public string LastName { get; set; }

            public int Age { get; set; }

            public override string ToString()
            {
                return FirstName + ", " + LastName + ", " + Age.ToString();
            }
        }

        static void Main(string[] args)
        {
            Console.WriteLine("Create the sync QLDB driver");
            IQldbDriver driver = QldbDriver.Builder()
                .WithLedger("quick-start")
                .WithSerializer(new ObjectSerializer())
                .Build();

            Console.WriteLine("Creating the table and index");

            // Creates the table and the index in the same transaction.
```

```
            // Note: Any code within the lambda can potentially execute multiple
 times due to retries.
            // For more information, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-retry-policy
            driver.Execute(txn =>
            {
                txn.Execute("CREATE TABLE Person");
                txn.Execute("CREATE INDEX ON Person(firstName)");
            });

            Console.WriteLine("Inserting a document");

            Person myPerson = new Person {
                FirstName = "John",
                LastName = "Doe",
                Age = 32
            };

            driver.Execute(txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("INSERT INTO Person ?",
 myPerson);
                txn.Execute(myQuery);
            });

            Console.WriteLine("Querying the table");

            // The result from driver.Execute() is buffered into memory because once
 the
            // transaction is committed, streaming the result is no longer possible.
            IResult<Person> selectResult = driver.Execute(txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person
 WHERE FirstName = ?", "John");
                return txn.Execute(myQuery);
            });

            foreach (Person person in selectResult)
            {
                Console.WriteLine(person);
                // John, Doe, 32
            }

            Console.WriteLine("Updating the document");
```

```
            driver.Execute(txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("UPDATE Person SET Age
 = ? WHERE FirstName = ?", 42, "John");
                txn.Execute(myQuery);
            });

            Console.WriteLine("Querying the table for the updated document");

            IResult<Person> updateResult = driver.Execute(txn =>
            {
                IQuery<Person> myQuery = txn.Query<Person>("SELECT * FROM Person
 WHERE FirstName = ?", "John");
                return txn.Execute(myQuery);
            });

            foreach (Person person in updateResult)
            {
                Console.WriteLine(person);
                // John, Doe, 42
            }

        }
    }
}
```

Verwendung der Ion-Bibliothek

Async

```
using System;
using System.Threading.Tasks;
using Amazon.IonDotnet.Tree;
using Amazon.IonDotnet.Tree.Impl;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
    {
```

```
        static IValueFactory valueFactory = new ValueFactory();

        static async Task Main(string[] args)
        {
            Console.WriteLine("Create the async QLDB driver");
            IAsyncQldbDriver driver = AsyncQldbDriver.Builder()
                .WithLedger("quick-start")
                .Build();

            Console.WriteLine("Creating the table and index");

            // Creates the table and the index in the same transaction.
            // Note: Any code within the lambda can potentially execute multiple
 times due to retries.
            // For more information, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-retry-policy
            await driver.Execute(async txn =>
            {
                await txn.Execute("CREATE TABLE Person");
                await txn.Execute("CREATE INDEX ON Person(firstName)");
            });

            Console.WriteLine("Inserting a document");

            // This is one way of creating Ion values. We can also use an IonLoader.
            // For more details, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-cookbook-dotnet.html#cookbook-dotnet.ion
            IIonValue ionPerson = valueFactory.NewEmptyStruct();
            ionPerson.SetField("firstName", valueFactory.NewString("John"));
            ionPerson.SetField("lastName", valueFactory.NewString("Doe"));
            ionPerson.SetField("age", valueFactory.NewInt(32));

            await driver.Execute(async txn =>
            {
                await txn.Execute("INSERT INTO Person ?", ionPerson);
            });

            Console.WriteLine("Querying the table");

            IIonValue ionFirstName = valueFactory.NewString("John");

            // The result from driver.Execute() is buffered into memory because once
 the
            // transaction is committed, streaming the result is no longer possible.
```

```
            IAsyncResult selectResult = await driver.Execute(async txn =>
            {
                return await txn.Execute("SELECT * FROM Person WHERE firstName = ?",
 ionFirstName);
            });

            await foreach (IIonValue row in selectResult)
            {
                Console.WriteLine(row.GetField("firstName").StringValue);
                Console.WriteLine(row.GetField("lastName").StringValue);
                Console.WriteLine(row.GetField("age").IntValue);
            }

            Console.WriteLine("Updating the document");

            IIonValue ionIntAge = valueFactory.NewInt(42);
            IIonValue ionFirstName2 = valueFactory.NewString("John");

            await driver.Execute(async txn =>
            {
                await txn.Execute("UPDATE Person SET age = ? WHERE firstName = ?",
 ionIntAge, ionFirstName2);
            });

            Console.WriteLine("Querying the table for the updated document");

            IIonValue ionFirstName3 = valueFactory.NewString("John");

            IAsyncResult updateResult = await driver.Execute(async txn =>
            {
                return await txn.Execute("SELECT * FROM Person WHERE firstName = ?",
 ionFirstName3);
            });

            await foreach (IIonValue row in updateResult)
            {
                Console.WriteLine(row.GetField("firstName").StringValue);
                Console.WriteLine(row.GetField("lastName").StringValue);
                Console.WriteLine(row.GetField("age").IntValue);
            }
        }
    }
}
```

Sync

```
using System;
using Amazon.IonDotnet.Tree;
using Amazon.IonDotnet.Tree.Impl;
using Amazon.QLDB.Driver;

namespace Amazon.QLDB.QuickStartGuide
{
    class Program
    {
        static IValueFactory valueFactory = new ValueFactory();

        static void Main(string[] args)
        {
            Console.WriteLine("Create the sync QLDB Driver");
            IQldbDriver driver = QldbDriver.Builder()
                .WithLedger("quick-start")
                .Build();

            Console.WriteLine("Creating the tables and index");

            // Creates the table and the index in the same transaction.
            // Note: Any code within the lambda can potentially execute multiple
 times due to retries.
            // For more information, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-retry-policy
            driver.Execute(txn =>
            {
                txn.Execute("CREATE TABLE Person");
                txn.Execute("CREATE INDEX ON Person(firstName)");
            });

            Console.WriteLine("Inserting a document");

            // This is one way of creating Ion values. We can also use an IonLoader.
            // For more details, see: https://docs.aws.amazon.com/qldb/latest/
developerguide/driver-cookbook-dotnet.html#cookbook-dotnet.ion
            IIonValue ionPerson = valueFactory.NewEmptyStruct();
            ionPerson.SetField("firstName", valueFactory.NewString("John"));
            ionPerson.SetField("lastName", valueFactory.NewString("Doe"));
            ionPerson.SetField("age", valueFactory.NewInt(32));

            driver.Execute(txn =>
```

```
            {
                txn.Execute("INSERT INTO Person ?", ionPerson);
            });

            Console.WriteLine("Querying the table");

            IIonValue ionFirstName = valueFactory.NewString("John");

            // The result from driver.Execute() is buffered into memory because once
the
            // transaction is committed, streaming the result is no longer possible.
            IResult selectResult = driver.Execute(txn =>
            {
                return txn.Execute("SELECT * FROM Person WHERE firstName = ?",
ionFirstName);
            });

            foreach (IIonValue row in selectResult)
            {
                Console.WriteLine(row.GetField("firstName").StringValue);
                Console.WriteLine(row.GetField("lastName").StringValue);
                Console.WriteLine(row.GetField("age").IntValue);
            }

            Console.WriteLine("Updating a document");

            IIonValue ionIntAge = valueFactory.NewInt(42);
            IIonValue ionFirstName2 = valueFactory.NewString("John");

            driver.Execute(txn =>
            {
                txn.Execute("UPDATE Person SET age = ? WHERE firstName = ?",
ionIntAge, ionFirstName2);
            });

            Console.WriteLine("Querying the table for the updated document");

            IIonValue ionFirstName3 = valueFactory.NewString("John");

            IResult updateResult = driver.Execute(txn =>
            {
                return txn.Execute("SELECT * FROM Person WHERE firstName = ?",
ionFirstName3);
            });
```

```
            foreach (IIonValue row in updateResult)
            {
                Console.WriteLine(row.GetField("firstName").StringValue);
                Console.WriteLine(row.GetField("lastName").StringValue);
                Console.WriteLine(row.GetField("age").IntValue);
            }
        }
    }
}
```

Um die vollständige Anwendung auszuführen, geben Sie den folgenden Befehl aus dem übergeordneten Verzeichnis Ihres `Amazon.QLDB.QuickStartGuide`-Projektverzeichnisses ein.

```
$ dotnet run --project Amazon.QLDB.QuickStartGuide
```

Amazon QLDB-Treiber für.NET — Kochbuch-Referenz

Dieses Referenzhandbuch zeigt häufige Anwendungsfälle des Amazon QLDB-Treibers für.NET. Es enthält C#-Codebeispiele, das zeigt, zeigt, wie Sie den Treiber verwenden, um grundlegende CRUD-Vorgänge (Erstellen, Lesen, Aktualisieren, Löschen) auszuführen. Es enthält auch Codebeispiele für die Verarbeitung von Amazon Ion-Daten. Darüber hinaus werden in diesem Leitfaden bewährte Verfahren vorgestellt, um Transaktionen idempotent zu machen und Eindeutigkeitsbeschränkungen zu implementieren.

> **Note**
>
> Dieses Thema enthält Codebeispiele für die standardmäßige Verarbeitung von Amazon Ion-Daten mit dem Ion-Objektmapper. QLDB führte den Ion Object Mapper in Version 1.3.0 des .NET-Treibers ein. Gegebenenfalls enthält dieses Thema auch Codebeispiele, bei denen die Standard-Ion-Bibliothek als Alternative verwendet wird. Weitere Informationen hierzu finden Sie unter Arbeiten mit Amazon Ion-Funktion.

Inhalt

Importieren des Treibers

Das folgende Codebeispiel importiert den Treiber.

```
using Amazon.QLDB.Driver;
using Amazon.QLDB.Driver.Generic;
using Amazon.QLDB.Driver.Serialization;
```

Verwendung der Ion-Bibliothek

```
using Amazon.QLDB.Driver;
using Amazon.IonDotnet.Builders;
```

Instantiieren des Treibers

Das folgende Codebeispiel erstellt eine Instanz des Treibers, die mithilfe der Standardeinstellungen eine Verbindung zu einem angegebenen Ledger-Namen herstellt.

Async

```
IAsyncQldbDriver driver = AsyncQldbDriver.Builder()
    .WithLedger("vehicle-registration")
    // Add Serialization library
    .WithSerializer(new ObjectSerializer())
    .Build();
```

Sync

```
IQldbDriver driver = QldbDriver.Builder()
    .WithLedger("vehicle-registration")
    // Add Serialization library
    .WithSerializer(new ObjectSerializer())
    .Build();
```

Verwendung der Ion-Bibliothek

Async

```
IAsyncQldbDriver driver = AsyncQldbDriver.Builder().WithLedger("vehicle-
registration").Build();
```

Sync

```
IQldbDriver driver = QldbDriver.Builder().WithLedger("vehicle-
registration").Build();
```

CRUD-Operationen

QLDB führt CRUD-Vorgänge (Erstellen, Lesen, Aktualisieren, Löschen) als Teil einer Transaktion durch.

> **Warning**
>
> Als bewährte Methode verwenden Sie Ihre Schreibtransaktionen für eine strikte idempotente.

Transaktionen idempotent machen

Wir empfehlen, Schreibtransaktionen idempotent zu machen, um unerwartete Nebenwirkungen bei Wiederholungsversuchen zu vermeiden. Eine Transaktion ist idempotent, wenn sie mehrfach ausgeführt werden kann und jedes Mal zu identischen Ergebnissen führt.

Stellen Sie sich zum Beispiel eine Transaktion vor, bei der ein Dokument in eine Tabelle mit dem Namen eingefügt wird`Person`. Die Transaktion sollte zunächst prüfen, ob das Dokument bereits in der Tabelle vorhanden ist. Ohne diese Überprüfung enthält die Tabelle möglicherweise doppelte Dokumente.

Nehmen wir an, QLDB führt die Transaktion auf der Serverseite erfolgreich durch, aber der Client läuft beim Warten auf eine Antwort ein Timeout. Wenn die Transaktion nicht idempotent ist, kann dasselbe Dokument im Falle eines erneuten Versuchs mehrmals eingefügt werden.

Verwendung von Indizes zur Vermeidung vollständiger Tabellenscans

Wir empfehlen außerdem, Anweisungen mit einer`WHERE` Prädikatklausel auszuführen, indem Sie einen Gleichheitsoperator für ein indiziertes Feld oder eine Dokument-ID verwenden, z. B.`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`. Ohne diese indizierte Suche muss QLDB einen Tabellenscan durchführen, was zu Transaktions-Timeouts oder OCC-Konflikten (Optimist Concurrency Control) führen kann.

Weitere Informationen zu OCC finden Sie unter Amazon QLDB-Nebenläufigkeit von Amazon QLDB-Nebenläufigkeit.

Implizit erstellte Transaktionen

Die Methode Amazon.QLDB.Driver.IQldbDriver .Execute akzeptiert eine Lambda-Funktion, die eine Instanz von Amazon.QLDB.Driver empfängt. TransactionExecutor, mit dem Sie Anweisungen ausführen können. Die Instanz von`TransactionExecutor` umschließt eine implizit erstellte Transaktion.

Sie können Anweisungen innerhalb der Lambda-Funktion ausführen, indem Sie die`Execute` Methode des Transaction Executors verwenden. Der Treiber schreibt die Transaktion implizit fest, wenn die Lambda-Funktion zurückkehrt.

In den folgenden Abschnitten wird gezeigt, wie grundlegende CRUD-Operationen ausgeführt, eine benutzerdefinierte Wiederholungslogik angegeben und Eindeutigkeitsbeschränkungen implementiert werden.

Inhalt

Erstellen von Tabellen

Async

```
IAsyncResult<Table> createResult = await driver.Execute(async txn =>
{
    IQuery<Table> query = txn.Query<Table>("CREATE TABLE Person");
    return await txn.Execute(query);
});

await foreach (var result in createResult)
{
    Console.WriteLine("{ tableId: " + result.TableId + " }");
    // The statement returns the created table ID:
    // { tableId: 4o5Uk09OcjC6PpJpLahceE }
}
```

Sync

```
IResult<Table> createResult = driver.Execute( txn =>
{
    IQuery<Table> query = txn.Query<Table>("CREATE TABLE Person");
    return txn.Execute(query);
});

foreach (var result in createResult)
{
```

```
    Console.WriteLine("{ tableId: " + result.TableId + " }");
    // The statement returns the created table ID:
    // { tableId: 4o5Uk09OcjC6PpJpLahceE }
}
```

Verwendung der Ion-Bibliothek

Async

```
// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("CREATE TABLE Person");
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the created table ID:
    // {
    //    tableId: "4o5Uk09OcjC6PpJpLahceE"
    // }
}
```

Sync

```
// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IResult result = driver.Execute(txn =>
{
    return txn.Execute("CREATE TABLE Person");
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the created table ID:
    // {
    //    tableId: "4o5Uk09OcjC6PpJpLahceE"
    // }
}
```

Erstellen von Indizes

Async

```
IAsyncResult<Table> createResult = await driver.Execute(async txn =>
{
    IQuery<Table> query = txn.Query<Table>("CREATE INDEX ON Person(firstName)");
    return await txn.Execute(query);
});

await foreach (var result in createResult)
{
    Console.WriteLine("{ tableId: " + result.TableId + " }");
    // The statement returns the updated table ID:
    // { tableId: 4o5Uk09OcjC6PpJpLahceE }
}
```

Sync

```
IResult<Table> createResult = driver.Execute(txn =>
{
    IQuery<Table> query = txn.Query<Table>("CREATE INDEX ON Person(firstName)");
    return txn.Execute(query);
});

foreach (var result in createResult)
{
    Console.WriteLine("{ tableId: " + result.TableId + " }");
    // The statement returns the updated table ID:
    // { tableId: 4o5Uk09OcjC6PpJpLahceE }
}
```

Verwendung der Ion-Bibliothek

Async

```
IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("CREATE INDEX ON Person(GovId)");
});

await foreach (IIonValue row in result)
```

```
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the updated table ID:
    // {
    //    tableId: "4o5Uk09OcjC6PpJpLahceE"
    // }
}
```

Sync

```
IResult result = driver.Execute(txn =>
{
    return txn.Execute("CREATE INDEX ON Person(GovId)");
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the updated table ID:
    // {
    //    tableId: "4o5Uk09OcjC6PpJpLahceE"
    // }
}
```

Dokumente lesen

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }
// Person class is defined as follows:
// public class Person
// {
//     public string GovId { get; set; }
//     public string FirstName { get; set; }
//  }

IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE GovId =
 'TOYENC486FH'"));
});

await foreach (Person person in result)
```

```
{
    Console.WriteLine(person.GovId); // Prints TOYENC486FH.
    Console.WriteLine(person.FirstName); // Prints Brent.
}
```

> **Note**
>
> Wenn Sie eine Abfrage ohne eine indizierte Suche ausführen, wird ein vollständiger Tabellenscan aufgerufen. In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Abfragen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Verwenden von Abfrageparametern

Das folgende Codebeispiel verwendet einen Abfrageparameter vom Typ C#.

```
IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE FirstName
 = ?", "Brent"));
});

await foreach (Person person in result)
{
    Console.WriteLine(person.GovId); // Prints TOYENC486FH.
    Console.WriteLine(person.FirstName); // Prints Brent.
}
```

Das folgende Codebeispiel verwendet mehrere Abfrageparameter vom Typ C#.

```
IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE GovId = ?
 AND FirstName = ?", "TOYENC486FH", "Brent"));
});

await foreach (Person person in result)
{
    Console.WriteLine(person.GovId); // Prints TOYENC486FH.
```

```
    Console.WriteLine(person.FirstName); // Prints Brent.
}
```

Das folgende Codebeispiel verwendet ein Array von Abfrageparametern vom Typ C#.

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }
// { "GovId": "ROEE1C1AABH", "FirstName" : "Jim" }
// { "GovId": "YH844DA7LDB", "FirstName" : "Mary" }

string[] ids = {
    "TOYENC486FH",
    "ROEE1C1AABH",
    "YH844DA7LDB"
};

IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE GovId IN
 (?,?,?)", ids));
});

await foreach (Person person in result)
{
    Console.WriteLine(person.FirstName); // Prints Brent on first iteration.
    // Prints Jim on second iteration.
    // Prints Mary on third iteration.
}
```

Das folgende Codebeispiel verwendet eine C#-Liste als Wert.

```
// Assumes that Person table has document as follows:
// { "GovId": "TOYENC486FH",
//   "FirstName" : "Brent",
//   "Vehicles": [
//      { "Make": "Volkswagen",
//        "Model": "Golf"},
//      { "Make": "Honda",
//        "Model": "Civic"}
//   ]
// }
// Person class is defined as follows:
// public class Person
```

```
// {
//     public string GovId { get; set; }
//     public string FirstName { get; set; }
//     public List<Vehicle> Vehicles { get; set; }
// }
// Vehicle class is defined as follows:
// public class Vehicle
// {
//     public string Make { get; set; }
//     public string Model { get; set; }
// }

List<Vehicle> vehicles = new List<Vehicle>
{
    new Vehicle
    {
        Make = "Volkswagen",
        Model = "Golf"
    },
    new Vehicle
    {
        Make = "Honda",
        Model = "Civic"
    }
};

IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE Vehicles
 = ?", vehicles));
});

await foreach (Person person in result)
{
    Console.WriteLine("{");
    Console.WriteLine($"  GovId: {person.GovId},");
    Console.WriteLine($"  FirstName: {person.FirstName},");
    Console.WriteLine("  Vehicles: [");
    foreach (Vehicle vehicle in person.Vehicles)
    {
        Console.WriteLine("  {");
        Console.WriteLine($"    Make: {vehicle.Make},");
        Console.WriteLine($"    Model: {vehicle.Model},");
        Console.WriteLine("  },");
```

```
    }
    Console.WriteLine("  ]");
    Console.WriteLine("}");
    // Prints:
    // {
    //     GovId: TOYENC486FH,
    //     FirstName: Brent,
    //     Vehicles: [
    //     {
    //         Make: Volkswagen,
    //         Model: Golf
    //     },
    //     {
    //         Make: Honda,
    //         Model: Civic
    //     },
    //     ]
    // }
}
```

Verwendung der Ion-Bibliothek

Async

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE GovId = 'TOYENC486FH'");
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

Sync

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }
```

```
IResult result = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE GovId = 'TOYENC486FH'");
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

> **Note**
>
> Wenn Sie eine Abfrage ohne eine indizierte Suche ausführen, wird ein vollständiger Tabellenscan aufgerufen. In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Abfragen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Das folgende Codebeispiel verwendet einen Abfrageparameter vom Typ Ion.

Async

```
IValueFactory valueFactory = new ValueFactory();
IIonValue ionFirstName = valueFactory.NewString("Brent");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE FirstName = ?",
 ionFirstName);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

Sync

```
IValueFactory valueFactory = new ValueFactory();
IIonValue ionFirstName = valueFactory.NewString("Brent");

IResult result = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE FirstName = ?", ionFirstName);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

Das folgende Codebeispiel verwendet mehrere Abfrageparameter.

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
IIonValue ionFirstName = valueFactory.NewString("Brent");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE GovId = ? AND FirstName
 = ?", ionGovId, ionFirstName);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
IIonValue ionFirstName = valueFactory.NewString("Brent");

IResult result = driver.Execute(txn =>
{
```

```
    return txn.Execute("SELECT * FROM Person WHERE GovId = ? AND FirstName = ?",
 ionGovId, ionFirstName);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

Das folgende Codebeispiel verwendet eine Liste von Abfrageparametern.

Async

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }
// { "GovId": "ROEE1C1AABH", "FirstName" : "Jim" }
// { "GovId": "YH844DA7LDB", "FirstName" : "Mary" }

IIonValue[] ionIds = {
    valueFactory.NewString("TOYENC486FH"),
    valueFactory.NewString("ROEE1C1AABH"),
    valueFactory.NewString("YH844DA7LDB")
};

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE GovId IN (?,?,?)", ionIds);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent on
 first iteration.
                                                              // Prints Jim on
 second iteration.
                                                              // Prints Mary on
 third iteration.
}
```

Sync

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }
// { "GovId": "ROEE1C1AABH", "FirstName" : "Jim" }
// { "GovId": "YH844DA7LDB", "FirstName" : "Mary" }

IIonValue[] ionIds = {
    valueFactory.NewString("TOYENC486FH"),
    valueFactory.NewString("ROEE1C1AABH"),
    valueFactory.NewString("YH844DA7LDB")
};

IResult result = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE GovId IN (?,?,?)", ionIds);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent on
 first iteration.
                                                              // Prints Jim on
 second iteration.
                                                              // Prints Mary on
 third iteration.
}
```

Das folgende Codebeispiel verwendet eine Ionen-Liste als Wert. Weitere Informationen zur Verwendung verschiedener Ion-Typen finden Sie unterArbeiten mit Amazon Ion-Datentypen in Amazon QLDB.

Async

```
// Assumes that Person table has document as follows:
// { "GovId": "TOYENC486FH",
//   "FirstName" : "Brent",
//   "Vehicles": [
//      { "Make": "Volkswagen",
//        "Model": "Golf"},
//      { "Make": "Honda",
//        "Model": "Civic"}
```

```
//   ]
// }

IIonValue ionVehicle1 = valueFactory.NewEmptyStruct();
ionVehicle1.SetField("Make", valueFactory.NewString("Volkswagen"));
ionVehicle1.SetField("Model", valueFactory.NewString("Golf"));

IIonValue ionVehicle2 = valueFactory.NewEmptyStruct();
ionVehicle2.SetField("Make", valueFactory.NewString("Honda"));
ionVehicle2.SetField("Model", valueFactory.NewString("Civic"));

IIonValue ionVehicles =  valueFactory.NewEmptyList();
ionVehicles.Add(ionVehicle1);
ionVehicles.Add(ionVehicle2);

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE Vehicles = ?",
 ionVehicles);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // Prints:
    // {
    //     GovId: "TOYENC486FN",
    //     FirstName: "Brent",
    //     Vehicles: [
    //     {
    //         Make: "Volkswagen",
    //         Model: "Golf"
    //     },
    //     {
    //         Make: "Honda",
    //         Model: "Civic"
    //     }
    //     ]
    // }
}
```

Sync

```
// Assumes that Person table has document as follows:
// { "GovId": "TOYENC486FH",
//   "FirstName" : "Brent",
//   "Vehicles": [
//      { "Make": "Volkswagen",
//        "Model": "Golf"},
//      { "Make": "Honda",
//        "Model": "Civic"}
//   ]
// }

IIonValue ionVehicle1 = valueFactory.NewEmptyStruct();
ionVehicle1.SetField("Make", valueFactory.NewString("Volkswagen"));
ionVehicle1.SetField("Model", valueFactory.NewString("Golf"));

IIonValue ionVehicle2 = valueFactory.NewEmptyStruct();
ionVehicle2.SetField("Make", valueFactory.NewString("Honda"));
ionVehicle2.SetField("Model", valueFactory.NewString("Civic"));

IIonValue ionVehicles =  valueFactory.NewEmptyList();
ionVehicles.Add(ionVehicle1);
ionVehicles.Add(ionVehicle2);

IResult result = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE Vehicles = ?", ionVehicles);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // Prints:
    // {
    //     GovId: "TOYENC486FN",
    //     FirstName: "Brent",
    //     Vehicles: [
    //     {
    //         Make: "Volkswagen",
    //         Model: "Golf"
    //     },
    //     {
    //         Make: "Honda",
```

```
    //         Model: "Civic"
    //     }
    //     ]
    // }
}
```

Einfügen von Dokumenten

Das folgende Codebeispiel fügt Ion-Datentypen ein.

```
string govId = "TOYENC486FH";

Person person = new Person
{
    GovId = "TOYENC486FH",
    FirstName = "Brent"
};

await driver.Execute(async txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IAsyncResult<Person> result = await txn.Execute(txn.Query<Person>("SELECT * FROM
 Person WHERE GovId = ?", govId));

    // Check if there is a record in the cursor.
    int count = await result.CountAsync();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    await txn.Execute(txn.Query<Document>("INSERT INTO Person ?", person));
});
```

Verwendung der Ion-Bibliothek

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
```

```
IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("GovId", valueFactory.NewString("TOYENC486FH"));
ionPerson.SetField("FirstName", valueFactory.NewString("Brent"));

await driver.Execute(async txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IAsyncResult result = await txn.Execute("SELECT * FROM Person WHERE GovId = ?",
 ionGovId);

    // Check if there is a record in the cursor.
    int count = await result.CountAsync();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    await txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("GovId", valueFactory.NewString("TOYENC486FH"));
ionPerson.SetField("FirstName", valueFactory.NewString("Brent"));

driver.Execute(txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IResult result = txn.Execute("SELECT * FROM Person WHERE GovId = ?", ionGovId);

    // Check if there is a record in the cursor.
    int count = result.Count();
    if (count > 0)
    {
        // Document already exists, no need to insert
```

```
        return;
    }

    // Insert the document.
    txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

Diese Transaktion fügt ein Dokument in diePerson Tabelle ein. Vor dem Einfügen wird zunächst geprüft, ob das Dokument bereits in der Tabelle vorhanden ist. Diese Prüfung macht die Transaktion idempotenter Natur. Selbst wenn Sie diese Transaktion mehrmals ausführen, verursacht sie keine unbeabsichtigten Nebenwirkungen.

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für dasGovId Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertenGovId Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Einfügen mehrerer Dokumente in eine Anweisung

Um mehrere Dokumente mit einer einzigenINSERT Anweisung einzufügen, können Sie wie folgt einenList C#-Parameter an die Anweisung übergeben.

```
Person person1 = new Person
{
    FirstName = "Brent",
    GovId = "TOYENC486FH"
};

Person person2 = new Person
{
    FirstName = "Jim",
    GovId = "ROEE1C1AABH"
};

List<Person> people = new List<Person>();
people.Add(person1);
people.Add(person2);
```

```
IAsyncResult<Document> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Document>("INSERT INTO Person ?", people));
});

await foreach (Document row in result)
{
    Console.WriteLine("{ documentId: " + row.DocumentId + " }");
    // The statement returns the created documents' ID:
    // { documentId: 6BFt5eJQDFLBW2aR8LPw42 }
    // { documentId: K5Zrcb6N3gmIEHgGhwoyKF }
}
```

Verwendung der Ion-Bibliothek

Um mehrere Dokumente mit einer einzigenINSERT Anweisung einzufügen, können Sie wie folgt einen Parameter vom Typ Ion list an die Anweisung übergeben.

Async

```
IIonValue ionPerson1 = valueFactory.NewEmptyStruct();
ionPerson1.SetField("FirstName", valueFactory.NewString("Brent"));
ionPerson1.SetField("GovId", valueFactory.NewString("TOYENC486FH"));

IIonValue ionPerson2 = valueFactory.NewEmptyStruct();
ionPerson2.SetField("FirstName", valueFactory.NewString("Jim"));
ionPerson2.SetField("GovId", valueFactory.NewString("ROEE1C1AABH"));

IIonValue ionPeople = valueFactory.NewEmptyList();
ionPeople.Add(ionPerson1);
ionPeople.Add(ionPerson2);

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("INSERT INTO Person ?", ionPeople);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the created documents' ID:
    // {
    //     documentId: "6BFt5eJQDFLBW2aR8LPw42"
```

```
    // }
    //
    // {
    //     documentId: "K5Zrcb6N3gmIEHgGhwoyKF"
    // }
}
```

Sync

```
IIonValue ionPerson1 = valueFactory.NewEmptyStruct();
ionPerson1.SetField("FirstName", valueFactory.NewString("Brent"));
ionPerson1.SetField("GovId", valueFactory.NewString("TOYENC486FH"));

IIonValue ionPerson2 = valueFactory.NewEmptyStruct();
ionPerson2.SetField("FirstName", valueFactory.NewString("Jim"));
ionPerson2.SetField("GovId", valueFactory.NewString("ROEE1C1AABH"));

IIonValue ionPeople = valueFactory.NewEmptyList();
ionPeople.Add(ionPerson1);
ionPeople.Add(ionPerson2);

IResult result = driver.Execute(txn =>
{
    return txn.Execute("INSERT INTO Person ?", ionPeople);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the created documents' ID:
    // {
    //     documentId: "6BFt5eJQDFLBW2aR8LPw42"
    // }
    //
    // {
    //     documentId: "K5Zrcb6N3gmIEHgGhwoyKF"
    // }
}
```

Sie schließen den variablen Platzhalter (?) nicht in doppelte eckige Klammern (<<...>>) ein, wenn Sie eine Ion-Liste übergeben. In manuellen PartiQL-Anweisungen bezeichnen doppelte spitze Klammern eine ungeordnete Sammlung, die als Bag bezeichnet wird.

Dokumente aktualisieren

```
string govId = "TOYENC486FH";
string firstName = "John";

IAsyncResult<Document> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Document>("UPDATE Person SET FirstName = ? WHERE
 GovId = ?", firstName , govId));
});

await foreach (Document row in result)
{
    Console.WriteLine("{ documentId: " + row.DocumentId + " }");
    // The statement returns the updated document ID:
    // { documentId: Djg30Zoltqy5M4BFsA2jSJ }
}
```

Verwendung der Ion-Bibliothek

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
IIonValue ionFirstName = valueFactory.NewString("John");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("UPDATE Person SET FirstName = ? WHERE GovId = ?",
 ionFirstName , ionGovId);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the updated document ID:
    // {
    //     documentId: "Djg30Zoltqy5M4BFsA2jSJ"
    // }
}
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
```

```
IIonValue ionFirstName = valueFactory.NewString("John");

IResult result = driver.Execute(txn =>
{
    return txn.Execute("UPDATE Person SET FirstName = ? WHERE GovId = ?",
 ionFirstName , ionGovId);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the updated document ID:
    // {
    //     documentId: "Djg30Zoltqy5M4BFsA2jSJ"
    // }
}
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Dokumente löschen

```
string govId = "TOYENC486FH";

IAsyncResult<Document> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Document>("DELETE FROM Person WHERE GovId = ?",
 govId));
});

await foreach (Document row in result)
{
    Console.WriteLine("{ documentId: " + row.DocumentId + " }");
    // The statement returns the updated document ID:
    // { documentId: Djg30Zoltqy5M4BFsA2jSJ }
}
```

Verwendung der Ion-Bibliothek

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("DELETE FROM Person WHERE GovId = ?", ionGovId);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the deleted document ID:
    // {
    //     documentId: "Djg30Zoltqy5M4BFsA2jSJ"
    // }
}
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IResult result = driver.Execute(txn =>
{
    return txn.Execute("DELETE FROM Person WHERE GovId = ?", ionGovId);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the deleted document ID:
    // {
    //     documentId: "Djg30Zoltqy5M4BFsA2jSJ"
    // }
}
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Ausführen mehrerer Kontoauszüge in einer Transaktion

```
// This code snippet is intentionally trivial. In reality you wouldn't do this because
 you'd
// set your UPDATE to filter on vin and insured, and check if you updated something or
 not.
public static async Task<bool> InsureVehicle(IAsyncQldbDriver driver, string vin)
{
    return await driver.Execute(async txn =>
    {
        // Check if the vehicle is insured.
        Amazon.QLDB.Driver.Generic.IAsyncResult<Vehicle> result = await txn.Execute(
            txn.Query<Vehicle>("SELECT insured FROM Vehicles WHERE vin = ? AND insured
 = FALSE", vin));

        if (await result.CountAsync() > 0)
        {
            // If the vehicle is not insured, insure it.
            await txn.Execute(
                txn.Query<Document>("UPDATE Vehicles SET insured = TRUE WHERE vin = ?",
 vin));
            return true;
        }
        return false;
    });
}
```

Verwendung der Ion-Bibliothek

Async

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
```

```
public static async Task<bool> InsureVehicle(IAsyncQldbDriver driver, string vin)
{
    ValueFactory valueFactory = new ValueFactory();
    IIonValue ionVin = valueFactory.NewString(vin);

    return await driver.Execute(async txn =>
    {
        // Check if the vehicle is insured.
        Amazon.QLDB.Driver.IAsyncResult result = await txn.Execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
 ionVin);

        if (await result.CountAsync() > 0)
        {
            // If the vehicle is not insured, insure it.
            await txn.Execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", ionVin);
            return true;
        }
        return false;
    });
}
```

Wiederholungslogik

Hinweise zur integrierten Wiederholungslogik des Treibers finden Sie unterVerstehen der Wiederholungsrichtlinie mit dem Treiber in Amazon QLDB.

Implementierung von Eindeutigkeitseinschränkungen

QLDB unterstützt keine eindeutigen Indizes, aber Sie können dieses Verhalten in Ihrer Anwendung implementieren.

Angenommen, Sie möchten eine Eindeutigkeitsbeschränkung für das`GovId` Feld in der`Person` Tabelle implementieren. Dazu können Sie eine Transaktion schreiben, die Folgendes bewirkt:

1. Stellen Sie sicher, dass die Tabelle keine vorhandenen Dokumente mit einem bestimmten Wert enthält`GovId`.
2. Fügen Sie das Dokument ein, wenn die Assertion erfolgreich ist.

Besteht eine konkurrierende Transaktion gleichzeitig die Assertion, wird nur eine der Transaktionen erfolgreich abgeschlossen. Die andere Transaktion schlägt mit einer OCC-Konfliktexception fehl.

Das folgende Codebeispiel veranschaulicht, wie Sie diese Einheitseinschränkungen implementieren.

```
string govId = "TOYENC486FH";

Person person = new Person
{
    GovId = "TOYENC486FH",
    FirstName = "Brent"
};

await driver.Execute(async txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IAsyncResult<Person> result = await txn.Execute(txn.Query<Person>("SELECT * FROM
 Person WHERE GovId = ?", govId));

    // Check if there is a record in the cursor.
    int count = await result.CountAsync();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    await txn.Execute(txn.Query<Document>("INSERT INTO Person ?", person));
});
```

Verwendung der Ion-Bibliothek

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("GovId", valueFactory.NewString("TOYENC486FH"));
ionPerson.SetField("FirstName", valueFactory.NewString("Brent"));

await driver.Execute(async txn =>
```

```
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IAsyncResult result = await txn.Execute("SELECT * FROM Person WHERE GovId = ?",
 ionGovId);

    // Check if there is a record in the cursor.
    int count = await result.CountAsync();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    await txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("GovId", valueFactory.NewString("TOYENC486FH"));
ionPerson.SetField("FirstName", valueFactory.NewString("Brent"));

driver.Execute(txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IResult result = txn.Execute("SELECT * FROM Person WHERE GovId = ?", ionGovId);

    // Check if there is a record in the cursor.
    int count = result.Count();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertem`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Arbeiten mit Amazon Ion-Funktion

Es gibt mehrere Möglichkeiten, Amazon-Ion-Daten in QLDB zu verarbeiten. Sie können die Ion-Bibliothek verwenden, um Ionenwerte zu erstellen und zu ändern. Oder Sie können den Ion-Objektmapper verwenden, um einfache alte C#-CLR-Objekte (POCO) Ion-Werten zuzuordnen. Version 1.3.0 des QLDB-Treibers für.NET führt die Unterstützung für den Ion Object Mapper ein.

Die folgenden Abschnitte enthalten Codebeispiele für die Verarbeitung von Ion-Daten mit beiden Techniken.

Inhalt

Import des Ionen-Moduls

```
using Amazon.IonObjectMapper;
```

Verwendung der Ion-Bibliothek

```
using Amazon.IonDotnet.Builders;
```

Erstellen von Ion-Typen

Das folgende Codebeispiel veranschaulicht, wie Sie Ion-Werte aus C#-Objekten mithilfe des Ion-Objektmapper erstellen.

```
// Assumes that Person class is defined as follows:
// public class Person
```

```
// {
//     public string FirstName { get; set; }
//     public int Age { get; set; }
// }

// Initialize the Ion Object Mapper
IonSerializer ionSerializer = new IonSerializer();

// The C# object to be serialized
Person person = new Person
{
    FirstName = "John",
    Age = 13
};

// Serialize the C# object into stream using the Ion Object Mapper
Stream stream = ionSerializer.Serialize(person);

// Load will take in stream and return a datagram; a top level container of Ion values.
IIonValue ionDatagram = IonLoader.Default.Load(stream);

// To get the Ion value within the datagram, we call GetElementAt(0).
IIonValue ionPerson = ionDatagram.GetElementAt(0);

Console.WriteLine(ionPerson.GetField("firstName").StringValue);
Console.WriteLine(ionPerson.GetField("age").IntValue);
```

Verwendung der Ion-Bibliothek

Die folgenden Codebeispiele zeigen die zwei Möglichkeiten, Ionen-Werte mithilfe der Ion-Bibliothek zu erstellen.

Verwenden von **`ValueFactory`**

```
using Amazon.IonDotnet.Tree;
using Amazon.IonDotnet.Tree.Impl;

IValueFactory valueFactory = new ValueFactory();

IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("firstName", valueFactory.NewString("John"));
ionPerson.SetField("age", valueFactory.NewInt(13));
```

```
Console.WriteLine(ionPerson.GetField("firstName").StringValue);
Console.WriteLine(ionPerson.GetField("age").IntValue);
```

Verwenden von **`IonLoader`**

```
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;

// Load will take in Ion text and return a datagram; a top level container of Ion
 values.
IIonValue ionDatagram = IonLoader.Default.Load("{firstName: \"John\", age: 13}");

// To get the Ion value within the datagram, we call GetElementAt(0).
IIonValue ionPerson = ionDatagram.GetElementAt(0);

Console.WriteLine(ionPerson.GetField("firstName").StringValue);
Console.WriteLine(ionPerson.GetField("age").IntValue);
```

Einen Ion-Binärdump abrufen

```
// Initialize the Ion Object Mapper with Ion binary serialization format
IonSerializer ionSerializer = new IonSerializer(new IonSerializationOptions
{
    Format = IonSerializationFormat.BINARY
});

// The C# object to be serialized
Person person = new Person
{
    FirstName = "John",
    Age = 13
};

MemoryStream stream = (MemoryStream) ionSerializer.Serialize(person);
Console.WriteLine(BitConverter.ToString(stream.ToArray()));
```

Verwendung der Ion-Bibliothek

```
// ionObject is an Ion struct
MemoryStream stream = new MemoryStream();
using (var writer = IonBinaryWriterBuilder.Build(stream))
{
```

```
    ionObject.WriteTo(writer);
    writer.Finish();
}

Console.WriteLine(BitConverter.ToString(stream.ToArray()));
```

Einen Ion-Textdump abrufen

```
// Initialize the Ion Object Mapper
IonSerializer ionSerializer = new IonSerializer(new IonSerializationOptions
{
    Format = IonSerializationFormat.TEXT
});

// The C# object to be serialized
Person person = new Person
{
    FirstName = "John",
    Age = 13
};

MemoryStream stream = (MemoryStream) ionSerializer.Serialize(person);
Console.WriteLine(System.Text.Encoding.UTF8.GetString(stream.ToArray()));
```

Verwendung der Ion-Bibliothek

```
// ionObject is an Ion struct
StringWriter sw = new StringWriter();
using (var writer = IonTextWriterBuilder.Build(sw))
{
    ionObject.WriteTo(writer);
    writer.Finish();
}

Console.WriteLine(sw.ToString());
```

Weitere Informationen zur Arbeit mit Ion finden Sie in der Amazon Ion-Dokumentation unter GitHub. Weitere Codebeispiele für die Arbeit mit Ion in QLDB finden Sie unterArbeiten mit Amazon Ion-Datentypen in Amazon QLDB.

Amazon QLDB-Treiber für Go

Um mit Daten in Ihrem Ledger zu arbeiten, können Sie von Ihrer Go-Anwendung aus mithilfe einesAWS bereitgestellten Treibers eine Verbindung zu Amazon QLDB herstellen. In den folgenden Themen wird die ersten Schritte mit dem QLDB-Treiber für Go tun.

Themen

Ressourcen für Fahrer

Weitere Informationen zu den vom Go-Treiber unterstützten Funktionen finden Sie in den folgenden Ressourcen:

- API-Referenz: 3.x, 2.x, 1.x
- Treiber-Quellcode (GitHub)
- Amazon Ion Cookbook

Voraussetzungen

Bevor Sie mit dem QLDB-Treiber für Go beginnen, müssen Sie die folgenden Schritte durchführen:

1. Befolgen Sie die AWS-Einrichtungsanweisungen unter Zugreifen auf Amazon QLDB. Diese umfasst die folgenden Funktionen:

    1. Registrieren Sie sich beiAWS.
    2. Erstellen Sie einen Benutzer mit den entsprechenden QLDB-Berechtigungen.
    3. Erteilen programmgesteuerten Zugriffs für die Entwicklung.
2. (Optional) Installieren Sie eine integrierte Entwicklungsumgebung (IDE) Ihrer Wahl. Eine Liste der häufig verwendeten IDEs für Go finden Sie unter Editor-Plugins und IDEs auf der Go-Website.

3. Laden Sie eine der folgenden Versionen von Go von der Go-Downloadseite herunter und installieren Sie sie:
   - 1.15 oder später — QLDB-Treiber für Go v3
   - 1.14 — QLDB-Treiber für Go v1 oder v2
4. Konfigurieren Sie die Entwicklungsumgebung für das AWS SDK for Go:
   1. Richten Sie Ihre AWS-Anmeldeinformationen ein. Wir empfehlen die Erstellung einer freigegebenen Anmeldeinformationsdatei.

      Anweisungen finden Sie unter Angeben von Anmeldeinformationen im AWS SDK for GoEntwicklerhandbuch.
   2. Stellen Sie Ihre Standardeinstellung einAWS-Region. Weitere Informationen erhalten Sie unter Angeben vonAWS-Region.

      Eine vollständige Liste der verfügbaren Regionen finden Sie unter Amazon QLDB-Endpunkte und Kontingente in der Allgemeine AWS-Referenz.

Als Nächstes können Sie eine einfache Beispielanwendung einrichten und kurze Codebeispiele ausführen — oder Sie können den Treiber in einem vorhandenen Go-Projekt installieren.

- Um den QLDB-Treiber und denAWS SDK for Go in einem vorhandenen Projekt zu installieren, fahren Sie mit fortInstallation.
- Informationen zum Einrichten eines Projekts und zur Ausführung von kurzen Codebeispielen, die grundlegende Datentransaktionen in einem Ledger veranschaulichen, finden Sie unterSchnellstart-Tutorial.

Installation

Der QLDB-Treiber für Go ist Open Source im GitHub Repository awslabs/amazon-qldb-driver-go. QLDB unterstützt die folgenden Treiberversionen und ihre Go-Abhängigkeiten.

Treiberversion	Go-Version	Status	Datum der letzten Veröffentlichung
1.x	1.14 oder später	Produktionsfreigabe	16. Juni 2021

Treiberversion	Go-Version	Status	Datum der letzten Veröffentlichung
2.x	1.14 oder später	Produktionsfreigabe	21. Juli 2021
3.x	1.15 oder später	Produktionsfreigabe	10. November 2022

So installieren Sie den Treiber

1. Stellen Sie sicher, dass Ihr Projekt Go-Module verwendet, um Projektabhängigkeiten zu installieren.
2. Geben Sie in Ihrem Projektverzeichnis den folgendengo `get` Befehl ein.

   3.x

   ```
   $ go get -u github.com/awslabs/amazon-qldb-driver-go/v3/qldbdriver
   ```

   2.x

   ```
   $ go get -u github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver
   ```

Durch die Installation des Treibers werden auch seine Abhängigkeiten installiert, einschließlich der Pakete AWS SDK for Goor AWS SDK for Gov2 und Amazon Ion.

Kurze Codebeispiele für die Ausführung grundlegender Datentransaktionen in einem Ledger finden Sie unterKochbuchreferenz.

Amazon QLDB-Treiber für Go — Schnellstart-Tutorial

In diesem Tutorial erfahren Sie, wie eine einfache Anwendung mit der neuesten Version des Amazon-QLDB-Treibers für Go geschehen. Dieses Handbuch enthält Schritte zum Installieren des Treibers und kurze Codebeispiele für grundlegende CRUD-Vorgänge (Create, Read, Update und Delete).

Themen

- Voraussetzungen
- Schritt 1: Installieren des Treibers

Voraussetzungen

Bevor Sie beginnen, stellen Sie sicher, dass die folgende Voraussetzung erfüllt ist:

1. Füllen des TreibersVoraussetzungen für den Go aus, falls noch geschehen geschehen geschehen geschehen geschehen. Dazu gehören die RegistrierungAWS, die Gewährung von programmatischem Zugriff für die Entwicklung und die Installation von Go.
2. Ledger mit dem Namen `quick-start` erstellen.

   Informationen zum Erstellen eines Ledgers finden Sie unterGrundfunktionen für Amazon QLDB-Ledgers oderSchritt 1: Erstellen eines neuen Ledgers unter Erste Schritte mit der Konsole.

Schritt 1: Installieren des Treibers

Stellen Sie sicher, dass Ihr Projekt Go-Module verwendet, um Projektabhängigkeiten zu installieren.

Geben Sie in Ihrem Projektverzeichnis den folgendego `get` Befehl ein.

```
$ go get -u github.com/awslabs/amazon-qldb-driver-go/v3/qldbdriver
```

Durch die Installation des Treibers werden auch seine Abhängigkeiten installiert, einschließlich der Pakete AWS SDK for Gov2 und Amazon Ion.

Schritt 2: Import der Pakete

Importieren Sie die folgendenAWS Pakete.

```
import (
    "context"
    "fmt"

    "github.com/amzn/ion-go/ion"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/qldbSession"
    "github.com/awslabs/amazon-qldb-driver-go/v3/qldbdriver"
)
```

Schritt 3: Initialisieren des Treibers

Initialisieren Sie eine Instance des Treibers, der eine Verbindung mit dem Ledger `quick-start` herstellt.

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
  panic(err)
}

qldbSession := qldbsession.NewFromConfig(cfg, func(options *qldbsession.Options) {
    options.Region = "us-east-1"
})
driver, err := qldbdriver.New(
    "quick-start",
    qldbSession,
    func(options *qldbdriver.DriverOptions) {
        options.LoggerVerbosity = qldbdriver.LogInfo
    })
if err != nil {
    panic(err)
}

defer driver.Shutdown(context.Background())
```

> **Note**
>
> Ersetzen Sie in diesem Codebeispiel *us-east-1* durch den Ort,AWS-Region an dem Sie Ihr Ledger erstellt haben.

Schritt 4: Erstellen einer Tabelle und eines Index

Im folgenden Codebeispiel wird veranschaulicht, wie `CREATE TABLE`- und `CREATE INDEX`-Anweisungen ausgeführt werden.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    _, err := txn.Execute("CREATE TABLE People")
    if err != nil {
        return nil, err
    }

    // When working with QLDB, it's recommended to create an index on fields we're
 filtering on.
    // This reduces the chance of OCC conflict exceptions with large datasets.
    _, err = txn.Execute("CREATE INDEX ON People (firstName)")
    if err != nil {
        return nil, err
    }

    _, err = txn.Execute("CREATE INDEX ON People (age)")
    if err != nil {
        return nil, err
    }

    return nil, nil
})
if err != nil {
    panic(err)
}
```

Dieser Code erstellt eine Tabelle mit dem Namen`People` und Indizes für die`age` Felder`firstName` und in dieser Tabelle. Indizes sind erforderlich, um die Abfrageleistung zu optimieren und dabei zu helfen, Konfliktausnahmen für Optimist Concurrency Control (OCC) zu begrenzen.

Schritt 5: Einfügen eines -Dokuments

Die folgenden Codebeispiele veranschaulichen, wie eine`INSERT` Anweisung ausgeführt wird. QLDB unterstützt die PartiQL-Abfragesprache (SQL-kompatibel) und das Amazon Ion-Datenformat (Superset von JSON).

Verwenden des literalen PartiQL

Der folgende Code fügt ein Dokument mithilfe einer partiQL-Anweisung im Zeichenkettenformat in die`People` Tabelle ein.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("INSERT INTO People {'firstName': 'Jane', 'lastName': 'Doe',
 'age': 77}")
})
if err != nil {
    panic(err)
}
```

Verwenden der Ionen-Datentypen

Ähnlich wie beim integrierten JSON-Paket von Go können Sie Go-Datentypen von und zu Ion marshalen und demarshalen.

1. Nehmen wir an, Sie haben die folgende Go-Struktur benannt`Person`.

   ```
   type Person struct {
       FirstName string `ion:"firstName"`
       LastName  string `ion:"lastName"`
       Age       int    `ion:"age"`
   }
   ```

2. Erstellen Sie eine Instance von `Person`.

   ```
   person := Person{"John", "Doe", 54}
   ```

   Der Treiber stellt für Sie eine ionenkodierte Textdarstellung von`person` bereit.

   > **Important**
   >
   > Damit marshal und unmarshal ordnungsgemäß funktionieren, müssen die Feldnamen der Go-Datenstruktur exportiert werden (der erste Buchstabe wird großgeschrieben).

3. Übergeben Sie die`person` Instanz an die`Execute` Methode der Transaktion.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("INSERT INTO People ?", person)
})
if err != nil {
    panic(err)
}
```

In diesem Beispiel wird ein Fragezeichen (?) als Variablenplatzhalter verwendet, um die Dokumentinformationen an die Anweisung zu übergeben. Wenn Sie Platzhalter verwenden, müssen Sie einen ionenkodierten Textwert übergeben.

> **Tip**
>
> Um mehrere Dokumente mit einer einzigenINSERT Anweisung einzufügen, können Sie wie folgt einen Parameter vom Typ list an die Anweisung übergeben.
>
> ```
> // people is a list
> txn.Execute("INSERT INTO People ?", people)
> ```
>
> Sie schließen den variablen Platzhalter (?) nicht in doppelte eckige Klammern (<<...>>) ein, wenn Sie eine Liste übergeben. In manuellen PartiQL-Anweisungen bezeichnen doppelte spitze Klammern eine ungeordnete Sammlung, die als Bag bezeichnet wird.

Schritt 6: Abfragen des -Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine SELECT-Anweisung ausgeführt wird.

```
p, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    result, err := txn.Execute("SELECT firstName, lastName, age FROM People WHERE age =
 54")
    if err != nil {
        return nil, err
    }

    // Assume the result is not empty
```

```
    hasNext := result.Next(txn)
    if !hasNext && result.Err() != nil {
        return nil, result.Err()
    }

    ionBinary := result.GetCurrentData()

    temp := new(Person)
    err = ion.Unmarshal(ionBinary, temp)
    if err != nil {
        return nil, err
    }

    return *temp, nil
})
if err != nil {
    panic(err)
}

var returnedPerson Person
returnedPerson = p.(Person)

if returnedPerson != person {
    fmt.Print("Queried result does not match inserted struct")
}
```

In diesem Beispiel wird Ihr Dokument aus der`People` Tabelle abgefragt, es wird davon ausgegangen, dass die Ergebnismenge nicht leer ist, und gibt Ihr Dokument anhand des Ergebnisses zurück.

Schritt 7: Aktualisieren des -Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine UPDATE-Anweisung ausgeführt wird.

```
person.Age += 10

_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("UPDATE People SET age = ? WHERE firstName = ?", person.Age,
 person.FirstName)
})
if err != nil {
    panic(err)
```

```
}
```

Schritt 8: Das aktualisierte Dokument abfragen

Im folgenden Codebeispiel wird die`People` Tabelle von`firstName` abgefragt und alle Dokumente in der Ergebnismenge zurückgegeben.

```
p, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    result, err := txn.Execute("SELECT firstName, lastName, age FROM People WHERE
 firstName = ?", person.FirstName)
    if err != nil {
        return nil, err
    }

    var people []Person
    for result.Next(txn) {
        ionBinary := result.GetCurrentData()

        temp := new(Person)
        err = ion.Unmarshal(ionBinary, temp)
        if err != nil {
            return nil, err
        }

        people = append(people, *temp)
    }
    if result.Err() != nil {
        return nil, result.Err()
    }

    return people, nil
})
if err != nil {
    panic(err)
}

var people []Person
people = p.([]Person)

updatedPerson := Person{"John", "Doe", 64}
if people[0] != updatedPerson {
    fmt.Print("Queried result does not match updated struct")
```

```
}
```

Schritt 9: Löschen der Tabelle

Im folgenden Codebeispiel wird veranschaulicht, wie eine `DROP TABLE`-Anweisung ausgeführt wird.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("DROP TABLE People")
})
if err != nil {
    panic(err)
}
```

Ausführen der vollständigen Anwendung

Das folgende Codebeispiel ist die vollständige Version der Anwendung. Anstatt die vorherigen Schritte einzeln auszuführen, können Sie dieses Codebeispiel auch kopieren und von Anfang bis Ende ausführen. Diese Anwendung demonstriert einige grundlegende CRUD-Operationen für den Ledger namens `quick-start`.

> **Note**
>
> Bevor Sie diesen Code ausführen, stellen Sie sicher, dass Sie noch keine aktive Tabelle mit dem Namen `People` im `quick-start`-Ledger besitzen.

```
package main

import (
    "context"
    "fmt"

    "github.com/amzn/ion-go/ion"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/qldbsession"
    "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

func main() {
```

```
    awsSession := session.Must(session.NewSession(aws.NewConfig().WithRegion("us-
east-1")))
    qldbSession := qldbsession.New(awsSession)

    driver, err := qldbdriver.New(
        "quick-start",
        qldbSession,
        func(options *qldbdriver.DriverOptions) {
            options.LoggerVerbosity = qldbdriver.LogInfo
        })
    if err != nil {
        panic(err)
    }
    defer driver.Shutdown(context.Background())

    _, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
        _, err := txn.Execute("CREATE TABLE People")
        if err != nil {
            return nil, err
        }

        // When working with QLDB, it's recommended to create an index on fields we're
 filtering on.
        // This reduces the chance of OCC conflict exceptions with large datasets.
        _, err = txn.Execute("CREATE INDEX ON People (firstName)")
        if err != nil {
            return nil, err
        }

        _, err = txn.Execute("CREATE INDEX ON People (age)")
        if err != nil {
            return nil, err
        }

        return nil, nil
    })
    if err != nil {
        panic(err)
    }

    _, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
```

```
        return txn.Execute("INSERT INTO People {'firstName': 'Jane', 'lastName': 'Doe',
'age': 77}")
    })
    if err != nil {
        panic(err)
    }

    type Person struct {
        FirstName string `ion:"firstName"`
        LastName  string `ion:"lastName"`
        Age       int    `ion:"age"`
    }

    person := Person{"John", "Doe", 54}

    _, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
        return txn.Execute("INSERT INTO People ?", person)
    })
    if err != nil {
        panic(err)
    }

    p, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
        result, err := txn.Execute("SELECT firstName, lastName, age FROM People WHERE
age = 54")
        if err != nil {
            return nil, err
        }

        // Assume the result is not empty
        hasNext := result.Next(txn)
        if !hasNext && result.Err() != nil {
            return nil, result.Err()
        }

        ionBinary := result.GetCurrentData()

        temp := new(Person)
        err = ion.Unmarshal(ionBinary, temp)
        if err != nil {
            return nil, err
        }
```

```
        return *temp, nil
    })
    if err != nil {
        panic(err)
    }

    var returnedPerson Person
    returnedPerson = p.(Person)

    if returnedPerson != person {
        fmt.Print("Queried result does not match inserted struct")
    }

    person.Age += 10

    _, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
        return txn.Execute("UPDATE People SET age = ? WHERE firstName = ?", person.Age,
person.FirstName)
    })
    if err != nil {
        panic(err)
    }

    p, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
        result, err := txn.Execute("SELECT firstName, lastName, age FROM People WHERE
firstName = ?", person.FirstName)
        if err != nil {
            return nil, err
        }

        var people []Person
        for result.Next(txn) {
            ionBinary := result.GetCurrentData()

            temp := new(Person)
            err = ion.Unmarshal(ionBinary, temp)
            if err != nil {
                return nil, err
            }

            people = append(people, *temp)
```

```
        }
        if result.Err() != nil {
            return nil, result.Err()
        }

        return people, nil
    })
    if err != nil {
        panic(err)
    }

    var people []Person
    people = p.([]Person)

    updatedPerson := Person{"John", "Doe", 64}
    if people[0] != updatedPerson {
        fmt.Print("Queried result does not match updated struct")
    }

    _, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
        return txn.Execute("DROP TABLE People")
    })
    if err != nil {
        panic(err)
    }
}
```

Amazon QLDB-Treiber für Go — Kochbuch-Referenz

Dieses Referenzhandbuch zeigt häufige Anwendungsfälle des Amazon QLDB-Treibers für Go. Es enthält Go-Codebeispiele, die zeigen, wie CRUD-Operationen (Erstellen, Lesen, Aktualisieren, Löschen) verwendet werden. Es enthält auch Codebeispiele für die Verarbeitung von Amazon Ion-Daten. Darüber hinaus werden in diesem Leitfaden bewährte Verfahren vorgestellt, um Transaktionen idempotent zu machen und Eindeutigkeitsbeschränkungen zu implementieren.

> **Note**
>
> Gegebenenfalls haben einige Anwendungsfälle unterschiedliche Codebeispiele für jede unterstützte Hauptversion des QLDB-Treibers für Go.

Inhalt

Importieren des Treibers

Das folgende Codebeispiel importiert den Treiber und andere erforderlicheAWS Pakete.

3.x

```
import (

    "github.com/amzn/ion-go/ion"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/qldbSession"
    "github.com/awslabs/amazon-qldb-driver-go/v3/qldbdriver"
```

```
)
```

2.x

```
import (

    "github.com/amzn/ion-go/ion"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/qldbsession"
    "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)
```

> **Note**
>
> In diesem Beispiel wird auch das Amazon Ion-Paket (`amzn/ion-go/ion`) importiert. Sie benötigen dieses Paket, um Ion-Daten zu verarbeiten, wenn Sie einige Datenoperationen in dieser Referenz ausführen. Weitere Informationen hierzu finden Sie unter Arbeiten mit Amazon Ion-on.

Instanziieren des Treibers

Das folgende Codebeispiel erstellt eine Instanz des Treibers, die eine Verbindung zu einem angegebenen Ledger-Namen in einem bestimmten Verzeichnis herstelltAWS-Region.

3.x

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic(err)
}

qldbSession := qldbsession.NewFromConfig(cfg, func(options *qldbsession.Options) {
    options.Region = "us-east-1"
})
driver, err := qldbdriver.New(
  "vehicle-registration",
  qldbSession,
  func(options *qldbdriver.DriverOptions) {
```

```
    options.LoggerVerbosity = qldbdriver.LogInfo
})
if err != nil {
  panic(err)
}

defer driver.Shutdown(context.Background())
```

2.x

```
awsSession := session.Must(session.NewSession(aws.NewConfig().WithRegion("us-
east-1")))
qldbSession := qldbsession.New(awsSession)

driver, err := qldbdriver.New(
  "vehicle-registration",
  qldbSession,
  func(options *qldbdriver.DriverOptions) {
    options.LoggerVerbosity = qldbdriver.LogInfo
  })
if err != nil {
  panic(err)
}
```

CRUD-Operationen

QLDB führt CRUD-Operationen (Erstellen, Lesen, Aktualisieren, Löschen) als Teil einer Transaktion durch.

> **Warning**
>
> Als bewährte Methode sollten Sie Ihre Schreibtransaktionen strikt idempotent gestalten.

Transaktionen idempotent machen

Wir empfehlen, Schreibtransaktionen idempotent zu machen, um unerwartete Nebenwirkungen bei Wiederholungsversuchen zu vermeiden. Eine Transaktion ist idempotent, wenn sie mehrfach ausgeführt werden kann und jedes Mal zu identischen Ergebnissen führt.

Stellen Sie sich zum Beispiel eine Transaktion vor, bei der ein Dokument in eine Tabelle mit dem Namen eingefügt wird`Person`. Die Transaktion sollte zunächst prüfen, ob das Dokument bereits in der Tabelle vorhanden ist. Ohne diese Überprüfung enthält die Tabelle möglicherweise doppelte Dokumente.

Nehmen wir an, QLDB führt die Transaktion auf der Serverseite erfolgreich durch, aber der Client läuft beim Warten auf eine Antwort ein Timeout. Wenn die Transaktion nicht idempotent ist, kann dasselbe Dokument im Falle eines erneuten Versuchs mehrmals eingefügt werden.

Verwendung von Indizes zur Vermeidung vollständiger Tabellenscans

Wir empfehlen außerdem, Anweisungen mit einer`WHERE` Prädikatklausel auszuführen, indem Sie einen Gleichheitsoperator für ein indiziertes Feld oder eine Dokument-ID verwenden, z. B.`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`. Ohne diese indizierte Suche muss QLDB einen Tabellenscan durchführen, was zu Transaktions-Timeouts oder OCC-Konflikten (Optimist Concurrency Control) führen kann.

Weitere Informationen zu OCC finden Sie unter Amazon QLDB-Nebenläufigkeit von Amazon QLDB-Nebenläufigkeit.

Implizit erstellte Transaktionen

Die Funktion QldbDriver.execute akzeptiert eine Lambda-Funktion, die eine Instanz von Transaction empfängt, mit der Sie Anweisungen ausführen können. Die Instanz von`Transaction` umschließt eine implizit erstellte Transaktion.

Sie können Anweisungen innerhalb der Lambda-Funktion ausführen, indem Sie die`Transaction.Execute` Funktion verwenden. Der Treiber schreibt die Transaktion implizit fest, wenn die Lambda-Funktion zurückkehrt.

In den folgenden Abschnitten wird gezeigt, wie grundlegende CRUD-Operationen ausgeführt, eine benutzerdefinierte Wiederholungslogik angegeben und Eindeutigkeitsbeschränkungen implementiert werden.

Inhalt

Erstellen von Tabellen

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("CREATE TABLE Person")
})
```

Erstellen von Indizes

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("CREATE INDEX ON Person(GovId)")
})
```

Dokumente lesen

```
var decodedResult map[string]interface{}

// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName": "Brent" }
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  result, err := txn.Execute("SELECT * FROM Person WHERE GovId = 'TOYENC486FH'")
  if err != nil {
    return nil, err
  }
  for result.Next(txn) {
    ionBinary := result.GetCurrentData()
    err = ion.Unmarshal(ionBinary, &decodedResult)
    if err != nil {
      return nil, err
```

```
    }
    fmt.Println(decodedResult) // prints map[GovId: TOYENC486FH FirstName:Brent]
  }
  if result.Err() != nil {
    return nil, result.Err()
  }
  return nil, nil
})
if err != nil {
  panic(err)
}
```

Verwenden Sie dieses Abfrageparameter

Das folgende Codebeispiel verwendet einen Abfrageparameter vom systemeigenen Typ.

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("SELECT * FROM Person WHERE GovId = ?", "TOYENC486FH")
})
if err != nil {
  panic(err)
}
```

Das folgende Codebeispiel verwendet mehrere Abfrageparameter.

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("SELECT * FROM Person WHERE GovId = ? AND FirstName = ?",
 "TOYENC486FH", "Brent")
})
if err != nil {
  panic(err)
}
```

Das folgende Codebeispiel verwendet eine Liste von Abfrageparametern.

```
govIDs := []string{}{"TOYENC486FH", "ROEE1", "YH844"}

result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("SELECT * FROM Person WHERE GovId IN (?,?,?)", govIDs...)
```

```
})
if err != nil {
  panic(err)
}
```

> **Note**
>
> Wenn Sie eine Abfrage ohne eine indizierte Suche ausführen, wird ein vollständiger Tabellenscan aufgerufen. In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Abfragen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Einfügen von -Dokumenten

Das folgende Codebeispiel fügt native Datentypen ein.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  // Check if a document with a GovId of TOYENC486FH exists
  // This is critical to make this transaction idempotent
  result, err := txn.Execute("SELECT * FROM Person WHERE GovId = ?", "TOYENC486FH")
  if err != nil {
    return nil, err
  }
  // Check if there are any results
  if result.Next(txn) {
    // Document already exists, no need to insert
  } else {
    person := map[string]interface{}{
      "GovId": "TOYENC486FH",
      "FirstName": "Brent",
    }
    _, err = txn.Execute("INSERT INTO Person ?", person)
    if err != nil {
      return nil, err
    }
  }
  return nil, nil
})
```

Diese Transaktion fügt ein Dokument in die`Person` Tabelle ein. Vor dem Einfügen wird zunächst geprüft, ob das Dokument bereits in der Tabelle vorhanden ist. Diese Prüfung macht die Transaktion idempotenter Natur. Selbst wenn Sie diese Transaktion mehrmals ausführen, verursacht sie keine unbeabsichtigten Nebenwirkungen.

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertem`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Einfügen mehrerer Dokumente in eine Anweisung

Um mehrere Dokumente mit einer einzigenINSERT Anweisung einzufügen, können Sie wie folgt einen Parameter vom Typ list an die Anweisung übergeben.

```
// people is a list
txn.Execute("INSERT INTO People ?", people)
```

Sie schließen den variablen Platzhalter (?) nicht in doppelte eckige Klammern (`<<...>>`) ein, wenn Sie eine Liste übergeben. In manuellen PartiQL-Anweisungen bezeichnen doppelte spitze Klammern eine ungeordnete Sammlung, die als Bag bezeichnet wird.

Dokumente aktualisieren

Das folgende Codebeispiel verwendet native Datentypen.

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("UPDATE Person SET FirstName = ? WHERE GovId = ?", "John",
 "TOYENC486FH")
})
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertem`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Dokumente löschen

Das folgende Codebeispiel verwendet native Datentypen.

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("DELETE FROM Person WHERE GovId = ?", "TOYENC486FH")
})
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Ausführen mehrerer Kontoauszüge in einer Transaktion

```
// This code snippet is intentionally trivial. In reality you wouldn't do this because
 you'd
// set your UPDATE to filter on vin and insured, and check if you updated something or
 not.
func InsureCar(driver *qldbdriver.QLDBDriver, vin string) (bool, error) {
    insured, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {

        result, err := txn.Execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE", vin)
        if err != nil {
            return false, err
        }

        hasNext := result.Next(txn)
        if !hasNext && result.Err() != nil {
            return false, result.Err()
        }

        if hasNext {
            _, err = txn.Execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin)
            if err != nil {
                return false, err
```

```
            }
            return true, nil
        }
        return false, nil
    })
    if err != nil {
        panic(err)
    }

    return insured.(bool), err
}
```

Wiederholungslogik

Die`Execute` Treiberfunktion verfügt über einen integrierten Wiederholungsmechanismus, der die Transaktion erneut versucht, wenn eine wiederholbare Ausnahme auftritt (z. B. Timeouts oder OCC-Konflikte). Die maximale Anzahl an Wiederholungsversuchen und die Backoff-Strategie sind konfigurierbar.

Das Standardlimit für Wiederholungsversuche ist`4`, und die Standard-Backoff-Strategie basiert auf ExponentialBackoffStrategyeiner Basis von`10` Millisekunden. Sie können die Wiederholungsrichtlinie pro Treiberinstanz und auch pro Transaktion festlegen, indem Sie eine Instanz von verwenden RetryPolicy.

Das folgende Codebeispiel spezifiziert die Wiederholungslogik mit einem benutzerdefinierten Wiederholungslimit und einer benutzerdefinierten Backoff-Strategie für eine Instanz des Treibers.

```
import (
  "github.com/aws/aws-sdk-go/aws"
  "github.com/aws/aws-sdk-go/aws/session"
  "github.com/aws/aws-sdk-go/service/qldbsession"
  "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

func main() {
  awsSession := session.Must(session.NewSession(aws.NewConfig().WithRegion("us-
east-1")))
  qldbSession := qldbsession.New(awsSession)

  // Configuring retry limit to 2
  retryPolicy := qldbdriver.RetryPolicy{MaxRetryLimit: 2}
```

```
  driver, err := qldbdriver.New("test-ledger", qldbSession, func(options
 *qldbdriver.DriverOptions) {
    options.RetryPolicy = retryPolicy
  })
  if err != nil {
    panic(err)
  }

  // Configuring an exponential backoff strategy with base of 20 milliseconds
  retryPolicy = qldbdriver.RetryPolicy{
    MaxRetryLimit: 2,
    Backoff: qldbdriver.ExponentialBackoffStrategy{SleepBase: 20, SleepCap: 4000,
    }}

  driver, err = qldbdriver.New("test-ledger", qldbSession, func(options
 *qldbdriver.DriverOptions) {
    options.RetryPolicy = retryPolicy
  })
  if err != nil {
    panic(err)
  }
}
```

Das folgende Codebeispiel spezifiziert die Wiederholungslogik mit einem benutzerdefinierten Wiederholungslimit und einer benutzerdefinierten Backoff-Strategie für eine bestimmte anonyme Funktion. Die`SetRetryPolicy` Funktion überschreibt die Wiederholungsrichtlinie, die für die Treiberinstanz festgelegt ist.

```
import (
  "context"
  "github.com/aws/aws-sdk-go/aws"
  "github.com/aws/aws-sdk-go/aws/session"
  "github.com/aws/aws-sdk-go/service/qldbsession"
  "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

func main() {
  awsSession := session.Must(session.NewSession(aws.NewConfig().WithRegion("us-
east-1")))
  qldbSession := qldbsession.New(awsSession)

  // Configuring retry limit to 2
  retryPolicy1 := qldbdriver.RetryPolicy{MaxRetryLimit: 2}
```

```
  driver, err := qldbdriver.New("test-ledger", qldbSession, func(options
 *qldbdriver.DriverOptions) {
    options.RetryPolicy = retryPolicy1
  })
  if err != nil {
    panic(err)
  }

  // Configuring an exponential backoff strategy with base of 20 milliseconds
  retryPolicy2 := qldbdriver.RetryPolicy{
    MaxRetryLimit: 2,
    Backoff: qldbdriver.ExponentialBackoffStrategy{SleepBase: 20, SleepCap: 4000,
    }}

  // Overrides the retry policy set by the driver instance
  driver.SetRetryPolicy(retryPolicy2)

  driver.Execute(context.Background(), func(txn qldbdriver.Transaction) (interface{},
 error) {
    return txn.Execute("CREATE TABLE Person")
  })
}
```

Implementierung von Eindeutigkeitseinschränkungen

QLDB unterstützt keine eindeutigen Indizes, aber Sie können dieses Verhalten in Ihrer Anwendung implementieren.

Angenommen, Sie möchten eine Eindeutigkeitsbeschränkung für das`GovId` Feld in der`Person` Tabelle implementieren. Dazu können Sie eine Transaktion schreiben, die Folgendes bewirkt:

1. Stellen Sie sicher, dass die Tabelle keine vorhandenen Dokumente mit einem bestimmten Wert enthält`GovId`.
2. Fügen Sie das Dokument ein, wenn die Assertion erfolgreich ist.

Besteht eine konkurrierende Transaktion gleichzeitig die Assertion, wird nur eine der Transaktionen erfolgreich abgeschlossen. Die andere Transaktion schlägt mit einer OCC-Konfliktexception fehl.

Das folgende Codebeispiel zeigt, wie Sie diese Einmalungseinschränkungslogik implementieren.

```
govID := "TOYENC486FH"
```

```
document := map[string]interface{}{
  "GovId":     "TOYENC486FH",
  "FirstName": "Brent",
}

result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  // Check if doc with GovId = govID exists
  result, err := txn.Execute("SELECT * FROM Person WHERE GovId = ?", govID)
  if err != nil {
    return nil, err
  }
  // Check if there are any results
  if result.Next(txn) {
    // Document already exists, no need to insert
    return nil, nil
  }
  return txn.Execute("INSERT INTO Person ?", document)
})
if err != nil {
  panic(err)
}
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Arbeiten mit Amazon Ion-on

In den folgenden Abschnitten sehen Sie, wie Sie Ion-Daten verwenden, um Ion-Daten auszuführen.

Inhalt

Import des Ionen-Moduls

```
import "github.com/amzn/ion-go/ion"
```

Ion-Typen erstellen

Die Ion-Bibliothek für Go unterstützt derzeit das Document Object Model (DOM) nicht, sodass Sie keine Ion-Datentypen erstellen können. Sie können jedoch zwischen den nativen Go-Typen und der Ion-Binärdatei wechseln, wenn Sie mit QLDB arbeiten.

Ion-Binärdatei

```
aDict := map[string]interface{}{
  "GovId": "TOYENC486FH",
  "FirstName": "Brent",
}

ionBytes, err := ion.MarshalBinary(aDict)
if err != nil {
  panic(err)
}

fmt.Println(ionBytes) // prints [224 1 0 234 238 151 129 131 222 147 135 190 144 133 71
 111 118 73 100 137 70 105 114 115 116 78 97 109 101 222 148 138 139 84 79 89 69 78 67
 52 56 54 70 72 139 133 66 114 101 110 116]
```

Ion-Text

```
aDict := map[string]interface{}{
  "GovId": "TOYENC486FH",
  "FirstName": "Brent",
}

ionBytes, err := ion.MarshalText(aDict)
if err != nil {
  panic(err)
}

fmt.Println(string(ionBytes)) // prints {FirstName:"Brent",GovId:"TOYENC486FH"}
```

Weitere Informationen zu Ion finden Sie in der Amazon Ion-Dokumentation unter GitHub. Weitere Codebeispiele für die Arbeit mit Ion in QLDB finden Sie unterArbeiten mit Amazon Ion-Datentypen in Amazon QLDB.

Amazon QLDB-Treiber für Node.js

Um mit Daten in Ihrem Ledger zu arbeiten, können Sie von Ihrer Anwendung Node.js aus mithilfe eines bereitgestellten Treibers eine AWS Verbindung zu Amazon QLDB herstellen. In den folgenden Themen werden die ersten Schritte mit dem QLDB-Treiber für Node.js beschrieben.

Themen

Ressourcen für Treiber

Weitere Informationen zu den vom Node.js-Treiber unterstützten Funktionen finden Sie in den folgenden Ressourcen:

- API-Referenz: 3.x, 2.x, 1.x
- Treiber-Quellcode () GitHub
- Quellcode einer Beispielanwendung (GitHub)
- Beispiele für Amazon Ion-Code
- Erstellen Sie eine einfache CRUD-Operation und einen Datenstrom auf QLDB mit AWS Lambda (Blog) AWS

Voraussetzungen

Bevor Sie mit dem QLDB-Treiber für Node.js beginnen, müssen Sie Folgendes tun:

1. Befolgen Sie die AWS-Einrichtungsanweisungen unter Zugreifen auf Amazon QLDB. Diese umfasst die folgenden Funktionen:

    1. Melden Sie sich an für. AWS
    2. Erstellen Sie einen Benutzer mit den entsprechenden QLDB-Berechtigungen.
    3. Gewähren Sie programmatischen Zugriff für die Entwicklung.
2. Installieren Sie Node.js Version 14.x oder höher von der Downloadseite Node.js. (Frühere Versionen des Treibers unterstützen Node.js Version 10.x oder höher.)
3. Konfigurieren Sie Ihre Entwicklungsumgebung für das AWSSDK für JavaScript in Node.js:

    1. Richten Sie Ihre AWS-Anmeldeinformationen ein. Wir empfehlen, eine Datei mit gemeinsamen Anmeldeinformationen zu erstellen.

        Eine Anleitung dazu finden Sie im AWS SDK for JavaScriptEntwicklerhandbuch unter Laden von Anmeldeinformationen in Node.js aus der Datei mit den gemeinsamen Anmeldeinformationen.
    2. Stellen Sie Ihre Standardeinstellung einAWS-Region. Wie das geht, erfahren Sie unter Einstellen vonAWS-Region.

        Eine vollständige Liste der verfügbaren Regionen finden Sie unter Amazon QLDB-Endpunkte und Kontingente in der. Allgemeine AWS-Referenz

Als Nächstes können Sie die vollständige Tutorial-Beispielanwendung herunterladen — oder Sie können nur den Treiber in einem Node.js -Projekt installieren und kurze Codebeispiele ausführen.

- Um den QLDB-Treiber und das AWS SDK für JavaScript in Node.js in einem vorhandenen Projekt zu installieren, fahren Sie mit fort. Installation
- Informationen zum Einrichten eines Projekts und zur Ausführung von Kurzcodebeispielen, die grundlegende Datentransaktionen in einem Ledger demonstrieren, finden Sie unter. Erste-Schrit-T-T-TSchrit
- Ausführlichere Beispiele für Daten- und Verwaltungs-API-Operationen in der vollständigen Beispielanwendung des Tutorials finden Sie unter. Anleitung zu Node.js

Installation

QLDB unterstützt die folgenden Treiberversionen und ihre Node.js Abhängigkeiten.

Treiberversion	Node.js-Version	Status	Datum der letzten Veröffentlichung
1.x	10.x oder später	Produktionsfreigabe	5. Juni 2020
2.x	10.x oder später	Produktionsfreigabe	6. Mai 2021
3.x	14.x oder später	Produktionsfreigabe	10. November 2023

Um den QLDB-Treiber mit npm (dem Paketmanager Node.js) zu installieren, geben Sie den folgenden Befehl aus Ihrem Projektstammverzeichnis ein.

3.x

```
npm install amazon-qldb-driver-nodejs
```

2.x

```
npm install amazon-qldb-driver-nodejs@2.2.0
```

1.x

```
npm install amazon-qldb-driver-nodejs@1.0.0
```

Der Treiber hat Peer-Abhängigkeiten von den folgenden Paketen. Sie müssen diese Pakete auch als Abhängigkeiten in Ihrem Projekt installieren.

3.x

Modularer aggregierter QLDB-Client (Management-API)

```
npm install @aws-sdk/client-qldb
```

Modularer aggregierter QLDB-Sitzungsclient (Daten-API)

```
npm install @aws-sdk/client-qldb-session
```

Amazon Ion-Datenformat

```
npm install ion-js
```

Reine JavaScript Implementierung von `BigInt`

```
npm install jsbi
```

2.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Amazon Ion-Datenformat

```
npm install ion-js@4.0.0
```

Reine JavaScript Implementierung von `BigInt`

```
npm install jsbi@3.1.1
```

1.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Amazon Ion-Datenformat

```
npm install ion-js@4.0.0
```

Reine JavaScript Implementierung von `BigInt`

```
npm install jsbi@3.1.1
```

Verwenden des Treibers zum Herstellen einer Verbindung mit einem Ledger

Dann können Sie den Treiber importieren und mit ihm eine Verbindung zu einem Ledger herstellen. Das folgende TypeScript Codebeispiel zeigt, wie eine Treiberinstanz für einen bestimmten Ledgernamen und AWS-Region erstellt wird.

3.x

```
import { Agent } from 'https';
import { QLDBSessionClientConfig } from "@aws-sdk/client-qldb-session";
import { QldbDriver, RetryConfig  } from 'amazon-qldb-driver-nodejs';
import { NodeHttpHandlerOptions } from "@aws-sdk/node-http-handler";

const maxConcurrentTransactions: number = 10;
const retryLimit: number = 4;

//Reuse connections with keepAlive
const lowLevelClientHttpOptions: NodeHttpHandlerOptions = {
    httpAgent: new Agent({
      maxSockets: maxConcurrentTransactions
    })
};

const serviceConfigurationOptions: QLDBSessionClientConfig = {
    region: "us-east-1"
};

//Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
const retryConfig: RetryConfig = new RetryConfig(retryLimit);
const qldbDriver: QldbDriver = new QldbDriver("testLedger",
 serviceConfigurationOptions, lowLevelClientHttpOptions, maxConcurrentTransactions,
 retryConfig);

qldbDriver.getTableNames().then(function(tableNames: string[]) {
    console.log(tableNames);
});
```

2.x

```
import { Agent } from 'https';
import { QldbDriver, RetryConfig  } from 'amazon-qldb-driver-nodejs';

const maxConcurrentTransactions: number = 10;
const retryLimit: number = 4;

//Reuse connections with keepAlive
const agentForQldb: Agent = new Agent({
    keepAlive: true,
```

```
    maxSockets: maxConcurrentTransactions
});

const serviceConfigurationOptions = {
    region: "us-east-1",
    httpOptions: {
        agent: agentForQldb
    }
};

//Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
const retryConfig: RetryConfig = new RetryConfig(retryLimit);
const qldbDriver: QldbDriver = new QldbDriver("testLedger",
 serviceConfigurationOptions, maxConcurrentTransactions, retryConfig);

qldbDriver.getTableNames().then(function(tableNames: string[]) {
    console.log(tableNames);
});
```

1.x

```
import { Agent } from 'https';
import { QldbDriver } from 'amazon-qldb-driver-nodejs';

const poolLimit: number = 10;
const retryLimit: number = 4;

//Reuse connections with keepAlive
const agentForQldb: Agent = new Agent({
    keepAlive: true,
    maxSockets: poolLimit
});

const serviceConfigurationOptions = {
    region: "us-east-1",
    httpOptions: {
        agent: agentForQldb
    }
};

const qldbDriver: QldbDriver = new QldbDriver("testLedger",
 serviceConfigurationOptions, retryLimit, poolLimit);
```

```
qldbDriver.getTableNames().then(function(tableNames: string[]) {
    console.log(tableNames);
});
```

Kurze Codebeispiele für die Ausführung von Basisdatentransaktionen in einem Ledger finden Sie unter. Kochbuchreferenz — Referenz für

Empfehlungen zur Einrichtung

Verbindungen mit Keep-Alive wiederverwenden

QLDB Node.js Treiber v3

Der standardmäßige Node.js-HTTP/HTTPS-Agent erstellt eine neue TCP-Verbindung für jede neue Anforderung. Um die Kosten für den Aufbau einer neuen Verbindung zu vermeiden, verwendet AWS SDK for JavaScript Version 3 standardmäßig TCP-Verbindungen wieder. Weitere Informationen und Informationen zum Deaktivieren der Wiederverwendung von Verbindungen finden Sie unter Wiederverwenden von Verbindungen mit Keep-Alive in Node.js im Entwicklerhandbuch. AWS SDK for JavaScript

Wir empfehlen, die Standardeinstellung zu verwenden, um Verbindungen im QLDB-Treiber für Node.js wiederzuverwenden. Stellen Sie bei der Treiberinitialisierung die Low-Level-Client-HTTP-Option auf denselben Wert ein`maxSockets`, den Sie für festgelegt haben. `maxConcurrentTransactions`

Sehen Sie sich zum Beispiel den folgenden Code JavaScript an. TypeScript

JavaScript

```
const qldb = require('amazon-qldb-driver-nodejs');
const https = require('https');

//Replace this value as appropriate for your application
const maxConcurrentTransactions = 50;

const agentForQldb = new https.Agent({
    //Set this to the same value as `maxConcurrentTransactions`(previously called
 `poolLimit`)
    //Do not rely on the default value of `Infinity`
    "maxSockets": maxConcurrentTransactions
});
```

```
const lowLevelClientHttpOptions = {
    httpAgent: agentForQldb
}

let driver = new qldb.QldbDriver("testLedger", undefined, lowLevelClientHttpOptions,
 maxConcurrentTransactions);
```

TypeScript

```
import { Agent } from 'https';
import { NodeHttpHandlerOptions } from "@aws-sdk/node-http-handler";
import { QldbDriver } from 'amazon-qldb-driver-nodejs';

//Replace this value as appropriate for your application
const maxConcurrentTransactions: number = 50;

const agentForQldb: Agent = new Agent({
    //Set this to the same value as `maxConcurrentTransactions`(previously called
 `poolLimit`)
    //Do not rely on the default value of `Infinity`
    maxSockets: maxConcurrentTransactions
});

const lowLevelClientHttpOptions: NodeHttpHandlerOptions = {
    httpAgent: agentForQldb
};

let driver = new QldbDriver("testLedger", undefined, lowLevelClientHttpOptions,
 maxConcurrentTransactions);
```

QLDB Node.js Treiber v2

Der standardmäßige Node.js-HTTP/HTTPS-Agent erstellt eine neue TCP-Verbindung für jede neue Anforderung. Um die Kosten für den Aufbau einer neuen Verbindung zu vermeiden, empfehlen wir, eine bestehende Verbindung wiederzuverwenden.

Verwenden Sie eine der folgenden Optionen, um Verbindungen im QLDB-Treiber für Node.js wiederzuverwenden:

- Stellen Sie während der Treiberinitialisierung die folgenden Low-Level-Client-HTTP-Optionen ein:

- `keepAlive` – `true`
- `maxSockets`— Derselbe Wert, den Sie für festgelegt haben `maxConcurrentTransactions`

Sehen Sie sich zum Beispiel den folgenden JavaScript TypeScript Code an.

JavaScript

```
const qldb = require('amazon-qldb-driver-nodejs');
const https = require('https');

//Replace this value as appropriate for your application
const maxConcurrentTransactions = 50;

const agentForQldb = new https.Agent({
    "keepAlive": true,
    //Set this to the same value as `maxConcurrentTransactions`(previously called
 `poolLimit`)
    //Do not rely on the default value of `Infinity`
    "maxSockets": maxConcurrentTransactions
});

const serviceConfiguration = { "httpOptions": {
    "agent": agentForQldb
}};

let driver = new qldb.QldbDriver("testLedger", serviceConfiguration,
 maxConcurrentTransactions);
```

TypeScript

```
import { Agent } from 'https';
import { ClientConfiguration } from 'aws-sdk/clients/acm';
import { QldbDriver } from 'amazon-qldb-driver-nodejs';

//Replace this value as appropriate for your application
const maxConcurrentTransactions: number = 50;

const agentForQldb: Agent = new Agent({
    keepAlive: true,
    //Set this to the same value as `maxConcurrentTransactions`(previously called
 `poolLimit`)
    //Do not rely on the default value of `Infinity`
    maxSockets: maxConcurrentTransactions
```

```
});

const serviceConfiguration: ClientConfiguration = { httpOptions: {
    agent: agentForQldb
}};

let driver = new QldbDriver("testLedger", serviceConfiguration,
 maxConcurrentTransactions);
```

- Alternativ können Sie die `AWS_NODEJS_CONNECTION_REUSE_ENABLED` Umgebungsvariable auf setzen1. Weitere Informationen finden Sie unter Wiederverwenden von Verbindungen mit Keep-Alive in Node.js im AWS SDK for JavaScript Entwicklerhandbuch.

> **Note**
>
> Wenn Sie diese Umgebungsvariable festlegen, wirkt sich dies auf alle aus, die AWS-Services die verwenden. AWS SDK for JavaScript

Amazon QLDB-Treiber für Node.js — Schnellstart-Tutorial

In diesem Tutorial erfahren Sie, wie Sie eine einfache Anwendung mit dem Amazon-QLDB-Treiber für Node.js einrichten. Dieses Handbuch enthält Schritte zur Installation des Treibers sowie Kurz JavaScript - und TypeScript Codebeispiele für grundlegende CRUD-Operationen (Erstellen, Lesen, Aktualisieren und Löschen). Ausführlichere Beispiele, die diese Vorgänge in einer vollständigen Beispielanwendung veranschaulichen, finden Sie im Anleitung zu Node.js.

> **Note**
>
> Gegebenenfalls enthalten einige Schritte unterschiedliche Codebeispiele für jede unterstützte Hauptversion des QLDB-Treibers für Node.js.

Themen

Voraussetzungen

Bevor Sie beginnen, stellen Sie sicher, dass die folgende Voraussetzung erfüllt ist:

1. Führen Sie die Voraussetzungen für den Node.js-Treiber aus, sofern Sie dies noch nicht getan haben. Dazu gehören die Registrierung fürAWS, die Gewährung von programmatischem Zugriff für die Entwicklung und die Installation von Node.js.
2. Ledger mit dem Namen `quick-start` erstellen.

   Informationen zum Erstellen eines Ledgers finden Sie unterGrundfunktionen für Amazon QLDB-Ledgers oderSchritt 1: Erstellen eines neuen Ledgers unter Erste Schritte mit der Konsole.

Wenn Sie es verwenden TypeScript, müssen Sie auch die folgenden Einrichtungsschritte ausführen.

Verwenden TypeScript

Zur Installation TypeScript

1. Installieren Sie das TypeScript Paket. Der QLDB-Treiber läuft auf TypeScript 3.8.x.

   ```
   $ npm install --global typescript@3.8.0
   ```

2. Führen Sie nach der Installation des Pakets den folgenden Befehl aus, um sicherzustellen, dass der TypeScript Compiler installiert ist.

   ```
   $ tsc --version
   ```

Um den Code in den folgenden Schritten auszuführen, beachten Sie, dass Sie Ihre TypeScript Datei zunächst wie folgt in ausführbaren JavaScript Code transpilieren müssen.

```
$ tsc app.ts; node app.js
```

Schritt 1: Einrichten des Projekts

Richten Sie zuerst Ihr Node.js-Projekt ein.

1. Erstellen Sie einen Ordner für Ihre Anwendung.

```
$ mkdir myproject
$ cd myproject
```

2. Um Ihr Projekt zu initialisieren, geben Sie den folgenden npm-Befehl ein und beantworten Sie die Fragen, die während der Einrichtung gestellt werden. Sie können für die meisten Fragen Standardwerte verwenden.

```
$ npm init
```

3. Installieren Sie den Amazon QLDB-Treiber für Node.js.
   - Verwenden von Version 3.x 3.x 3.x

```
$ npm install amazon-qldb-driver-nodejs --save
```

   - Version 2.x verwenden

```
$ npm install amazon-qldb-driver-nodejs@2.2.0 --save
```

   - Version 1.x verwenden

```
$ npm install amazon-qldb-driver-nodejs@1.0.0 --save
```

4. Installieren Sie die Peer-Abhängigkeiten des Treibers.
   - Verwenden von Version 3.x 3.x 3.x

```
$ npm install @aws-sdk/client-qldb-session --save
$ npm install ion-js --save
$ npm install jsbi --save
```

   - Verwenden von Version 2.x oder 1.x

```
$ npm install aws-sdk --save
$ npm install ion-js@4.0.0 --save
$ npm install jsbi@3.1.1 --save
```

5. Erstellen Sie eine neue Datei mit dem Namen`app.js` für JavaScript oder`app.ts` für TypeScript.

   Fügen Sie dann schrittweise die Codebeispiele in den folgenden Schritten hinzu, um einige grundlegende CRUD-Operationen auszuprobieren. Oder Sie können das step-by-step Tutorial überspringen und stattdessen die komplette Anwendung ausführen.

Schritt 2: Initialisieren des Treibers

Initialisieren Sie eine Instance des Treibers, der eine Verbindung mit dem Ledger `quick-start` herstellt. Fügen Sie Ihrer`app.js`app.ts OR-Datei den folgenden Code hinzu.

Verwenden von Version 3.x 3.x 3.x

JavaScript

```
var qldb = require('amazon-qldb-driver-nodejs');
var https = require('https');

function main() {
    const maxConcurrentTransactions = 10;
    const retryLimit = 4;

    const agentForQldb = new https.Agent({
        maxSockets: maxConcurrentTransactions
    });

    const lowLevelClientHttpOptions = {
       httpAgent: agentForQldb
    }

    const serviceConfigurationOptions = {
        region: "us-east-1"
    };

    // Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
    var retryConfig = new qldb.RetryConfig(retryLimit);
    var driver = new qldb.QldbDriver("quick-start", serviceConfigurationOptions,
 lowlevelClientHttpOptions, maxConcurrentTransactions, retryConfig);
}

main();
```

TypeScript

```
import { Agent } from "https";
import { NodeHttpHandlerOptions } from "@aws-sdk/node-http-handler";
import { QLDBSessionClientConfig } from "@aws-sdk/client-qldb-session";
import { QldbDriver, RetryConfig } from "amazon-qldb-driver-nodejs";

function main(): void {
    const maxConcurrentTransactions: number = 10;
    const agentForQldb: Agent = new Agent({
        maxSockets: maxConcurrentTransactions
    });

    const lowLevelClientHttpOptions: NodeHttpHandlerOptions = {
      httpAgent: agentForQldb
    };

    const serviceConfigurationOptions: QLDBSessionClientConfig = {
        region: "us-east-1"
    };

    const retryLimit: number = 4;
    // Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
    const retryConfig: RetryConfig = new RetryConfig(retryLimit);
    const driver: QldbDriver = new QldbDriver("quick-start",
 serviceConfigurationOptions, lowLevelClientHttpOptions, maxConcurrentTransactions,
 retryConfig);
}

if (require.main === module) {
    main();
}
```

Note

- Ersetzen Sie in diesem Codebeispiel *us-east-1* durch den Ort,AWS-Region an dem Sie Ihr Ledger erstellt haben.
- Der Einfachheit halber verwenden die übrigen Codebeispiele in diesem Handbuch einen Treiber mit Standardeinstellungen, wie im folgenden Beispiel für Version 1.x

angegeben. Sie könnenRetryConfig stattdessen auch Ihre eigene Treiberinstanz mit einer benutzerdefinierten Instanz verwenden.

Version 2.x verwenden

JavaScript

```
var qldb = require('amazon-qldb-driver-nodejs');
var https = require('https');

function main() {
    var maxConcurrentTransactions = 10;
    var retryLimit = 4;

    var agentForQldb = new https.Agent({
        keepAlive: true,
        maxSockets: maxConcurrentTransactions
    });

    var serviceConfigurationOptions = {
        region: "us-east-1",
        httpOptions: {
            agent: agentForQldb
        }
    };

    // Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
    var retryConfig = new qldb.RetryConfig(retryLimit);
    var driver = new qldb.QldbDriver("quick-start", serviceConfigurationOptions,
 maxConcurrentTransactions, retryConfig);
}

main();
```

TypeScript

```
import { QldbDriver, RetryConfig } from "amazon-qldb-driver-nodejs";
import { ClientConfiguration } from "aws-sdk/clients/acm";
import { Agent } from "https";
```

```
function main(): void {
    const maxConcurrentTransactions: number = 10;
    const agentForQldb: Agent = new Agent({
        keepAlive: true,
        maxSockets: maxConcurrentTransactions
    });
    const serviceConfigurationOptions: ClientConfiguration = {
        region: "us-east-1",
        httpOptions: {
            agent: agentForQldb
        }
    };
    const retryLimit: number = 4;
    // Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
    const retryConfig: RetryConfig = new RetryConfig(retryLimit);
    const driver: QldbDriver = new QldbDriver("quick-start",
 serviceConfigurationOptions, maxConcurrentTransactions, retryConfig);
}

if (require.main === module) {
    main();
}
```

Note

- Ersetzen Sie in diesem Codebeispiel *us-east-1* durch den Ort,AWS-Region an dem Sie Ihr Ledger erstellt haben.
- Version 2.x führt den neuen optionalen Parameter`RetryConfig` für die Initialisierung ein`QldbDriver`.
- Der Einfachheit halber verwenden die übrigen Codebeispiele in diesem Handbuch einen Treiber mit Standardeinstellungen, wie im folgenden Beispiel für Version 1.x angegeben. Sie können`RetryConfig` stattdessen auch Ihre eigene Treiberinstanz mit einer benutzerdefinierten Instanz verwenden.
- Dieses Codebeispiel initialisiert einen Treiber, der bestehende Verbindungen wiederverwendet, indem Keep-Alive-Optionen gesetzt werden. Weitere Informationen finden Sie unter demEmpfehlungen zur Einrichtung Treiber Node.js.

Version 1.x verwenden

JavaScript

```
const qldb = require('amazon-qldb-driver-nodejs');

function main() {
    // Use default settings
    const driver = new qldb.QldbDriver("quick-start");
}

main();
```

TypeScript

```
import { QldbDriver } from "amazon-qldb-driver-nodejs";

function main(): void {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Sie können die`AWS_REGION` Umgebungsvariable festlegen, um die Region anzugeben. Weitere Informationen finden Sie AWS-RegionimAWS SDK for JavaScript Entwicklerhandbuch unter Setting the.

Schritt 3: Erstellen einer Tabelle und eines Index

Die folgenden Codebeispiele zeigen, wie `CREATE TABLE`- und `CREATE INDEX`-Anweisungen ausgeführt werden.

1. Fügen Sie die folgende Funktion hinzu, die eine Tabelle mit dem Namen `People` erstellt.

JavaScript

```
async function createTable(txn) {
    await txn.execute("CREATE TABLE People");
}
```

TypeScript

```
async function createTable(txn: TransactionExecutor): Promise<void> {
    await txn.execute("CREATE TABLE People");
}
```

2. Fügen Sie die folgende Funktion hinzu, die einen Index für das Feld `firstName` in der Tabelle `People` erstellt. Indizes sind erforderlich, um die Abfrageleistung zu optimieren und dabei zu helfen, Konfliktausnahmen für Optimist Concurrency Control (OCC) zu begrenzen.

   JavaScript

   ```
   async function createIndex(txn) {
       await txn.execute("CREATE INDEX ON People (firstName)");
   }
   ```

   TypeScript

   ```
   async function createIndex(txn: TransactionExecutor): Promise<void> {
       await txn.execute("CREATE INDEX ON People (firstName)");
   }
   ```

3. In der `main` Funktion rufen `createTable` Sie zuerst und dann an `createIndex`.

   JavaScript

   ```
   const qldb = require('amazon-qldb-driver-nodejs');

   async function main() {
       // Use default settings
       const driver = new qldb.QldbDriver("quick-start");

       await driver.executeLambda(async (txn) => {
           console.log("Create table People");
           await createTable(txn);
   ```

```
        console.log("Create index on firstName");
        await createIndex(txn);
    });

    driver.close();
}

main();
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";

async function main(): Promise<void> {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");

    await driver.executeLambda(async (txn: TransactionExecutor) => {
        console.log("Create table People");
        await createTable(txn);
        console.log("Create index on firstName");
        await createIndex(txn);
    });

    driver.close();
}

if (require.main === module) {
    main();
}
```

4. Führen Sie den Code aus, um die Tabelle und den Index zu erstellen.

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

Schritt 4: Einfügen eines Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine `INSERT`-Anweisung ausgeführt wird. QLDB unterstützt die PartiQL-Abfragesprache (SQL-kompatibel) und das Amazon Ion-Datenformat (Superset von JSON).

1. Fügen Sie die folgende Funktion hinzu, die ein Dokument in die Tabelle `People` einfügt.

   JavaScript

   ```
   async function insertDocument(txn) {
       const person = {
           firstName: "John",
           lastName: "Doe",
           age: 42
       };
       await txn.execute("INSERT INTO People ?", person);
   }
   ```

   TypeScript

   ```
   async function insertDocument(txn: TransactionExecutor): Promise<void> {
       const person: Record<string, any> = {
           firstName: "John",
           lastName: "Doe",
           age: 42
       };
       await txn.execute("INSERT INTO People ?", person);
   }
   ```

   In diesem Beispiel wird ein Fragezeichen (?) als Variablenplatzhalter verwendet, um die Dokumentinformationen an die Anweisung zu übergeben. Die`execute` Methode unterstützt Werte sowohl in den Amazon Ion-Typen als auch in den nativen Typen von Node.js.

   > **Tip**
   >
   > Um mehrere Dokumente mit einer einzigenINSERT Anweisung einzufügen, können Sie wie folgt einen Parameter vom Typ list an die Anweisung übergeben.
   >
   > ```
   > // people is a list
   > ```

```
txn.execute("INSERT INTO People ?", people);
```

Sie schließen den variablen Platzhalter (?) nicht in doppelte eckige Klammern (<<...>>) ein, wenn Sie eine Liste übergeben. In manuellen PartiQL-Anweisungen bezeichnen doppelte spitze Klammern eine ungeordnete Sammlung, die als Bag bezeichnet wird.

2. Entfernen Sie in der`main` Funktion die`createTable`und`createIndex` AND-Aufrufe und fügen Sie einen Anruf hinzu`insertDocument`.

JavaScript

```
const qldb = require('amazon-qldb-driver-nodejs');

async function main() {
    // Use default settings
    const driver = new qldb.QldbDriver("quick-start");

    await driver.executeLambda(async (txn) => {
        console.log("Insert document");
        await insertDocument(txn);
    });

    driver.close();
}

main();
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";

async function main(): Promise<void> {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");

    await driver.executeLambda(async (txn: TransactionExecutor) => {
        console.log("Insert document");
        await insertDocument(txn);
    });
```

```
    driver.close();
}

if (require.main === module) {
    main();
}
```

Schritt 5: Abfragen des Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine `SELECT`-Anweisung ausgeführt wird.

1. Fügen Sie die folgende Funktion hinzu, die ein Dokument aus der Tabelle `People` abfragt.

   JavaScript

   ```
   async function fetchDocuments(txn) {
       return await txn.execute("SELECT firstName, age, lastName FROM People WHERE
    firstName = ?", "John");
   }
   ```

   TypeScript

   ```
   async function fetchDocuments(txn: TransactionExecutor): Promise<dom.Value[]> {
       return (await txn.execute("SELECT firstName, age, lastName FROM People WHERE
    firstName = ?", "John")).getResultList();
   }
   ```

2. Fügen Sie in der`main` Funktion den folgenden Aufruf an`fetchDocuments` nach dem Aufruf von hinzu`insertDocument`.

   JavaScript

   ```
   const qldb = require('amazon-qldb-driver-nodejs');

   async function main() {
       // Use default settings
       const driver = new qldb.QldbDriver("quick-start");

       var resultList = await driver.executeLambda(async (txn) => {
           console.log("Insert document");
           await insertDocument(txn);
   ```

```
        console.log("Fetch document");
        var result = await fetchDocuments(txn);
        return result.getResultList();
    });

    // Pretty print the result list
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
}

main();
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { dom } from "ion-js";

async function main(): Promise<void> {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");

    const resultList: dom.Value[] = await driver.executeLambda(async (txn:
 TransactionExecutor) => {
        console.log("Insert document");
        await insertDocument(txn);
        console.log("Fetch document");
        return await fetchDocuments(txn);
    });

    // Pretty print the result list
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
}

if (require.main === module) {
    main();
}
```

Schritt 6: Aktualisieren des Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine UPDATE-Anweisung ausgeführt wird.

1. Fügen Sie die folgende Funktion hinzu, die ein Dokument in der `People`-Tabelle aktualisiert, indem Sie `lastName` in `"Stiles"` ändern.

   JavaScript

   ```
   async function updateDocuments(txn) {
       await txn.execute("UPDATE People SET lastName = ? WHERE firstName = ?",
    "Stiles", "John");
   }
   ```

   TypeScript

   ```
   async function updateDocuments(txn: TransactionExecutor): Promise<void> {
       await txn.execute("UPDATE People SET lastName = ? WHERE firstName = ?",
    "Stiles", "John");
   }
   ```

2. Fügen Sie in der`main` Funktion den folgenden Aufruf an`updateDocuments` nach dem Aufruf von hinzu`fetchDocuments`. Rufen Sie dann`fetchDocuments` erneut an, um die aktualisierten Ergebnisse zu sehen.

   JavaScript

   ```
   const qldb = require('amazon-qldb-driver-nodejs');

   async function main() {
       // Use default settings
       const driver = new qldb.QldbDriver("quick-start");

       var resultList = await driver.executeLambda(async (txn) => {
           console.log("Insert document");
           await insertDocument(txn);
           console.log("Fetch document");
           await fetchDocuments(txn);
           console.log("Update document");
           await updateDocuments(txn);
           console.log("Fetch document after update");
           var result = await fetchDocuments(txn);
           return result.getResultList();
       });

       // Pretty print the result list
   ```

```
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
}

main();
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { dom } from "ion-js";

async function main(): Promise<void> {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");

    const resultList: dom.Value[] = await driver.executeLambda(async (txn:
 TransactionExecutor) => {
        console.log("Insert document");
        await insertDocument(txn);
        console.log("Fetch document");
        await fetchDocuments(txn);
        console.log("Update document");
        await updateDocuments(txn);
        console.log("Fetch document after update");
        return await fetchDocuments(txn);
    });

    // Pretty print the result list
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
}

if (require.main === module) {
    main();
}
```

3. Führen Sie den Code aus, um ein Dokument einzufügen, abzufragen und zu aktualisieren.

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

Ausführen der vollständigen Anwendung

Die folgenden Codebeispiele sind die vollständigen Versionen von`app.js` und`app.ts`. Anstatt die vorherigen Schritte einzeln auszuführen, können Sie diesen Code auch von Anfang bis Ende ausführen. Diese Anwendung demonstriert einige grundlegende CRUD-Operationen für den Ledger namens `quick-start`.

> **Note**
>
> Bevor Sie diesen Code ausführen, stellen Sie sicher, dass Sie noch keine aktive Tabelle mit dem Namen `People` im `quick-start`-Ledger besitzen.

JavaScript

```
const qldb = require('amazon-qldb-driver-nodejs');

async function createTable(txn) {
    await txn.execute("CREATE TABLE People");
}

async function createIndex(txn) {
    await txn.execute("CREATE INDEX ON People (firstName)");
}

async function insertDocument(txn) {
    const person = {
        firstName: "John",
        lastName: "Doe",
        age: 42
    };
    await txn.execute("INSERT INTO People ?", person);
}

async function fetchDocuments(txn) {
```

```
    return await txn.execute("SELECT firstName, age, lastName FROM People WHERE
 firstName = ?", "John");
}

async function updateDocuments(txn) {
    await txn.execute("UPDATE People SET lastName = ? WHERE firstName = ?",
 "Stiles", "John");
}

async function main() {
    // Use default settings
    const driver = new qldb.QldbDriver("quick-start");

    var resultList = await driver.executeLambda(async (txn) => {
        console.log("Create table People");
        await createTable(txn);
        console.log("Create index on firstName");
        await createIndex(txn);
        console.log("Insert document");
        await insertDocument(txn);
        console.log("Fetch document");
        await fetchDocuments(txn);
        console.log("Update document");
        await updateDocuments(txn);
        console.log("Fetch document after update");
        var result = await fetchDocuments(txn);
        return result.getResultList();
    });

    // Pretty print the result list
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
}

main();
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { dom } from "ion-js";

async function createTable(txn: TransactionExecutor): Promise<void> {
    await txn.execute("CREATE TABLE People");
```

```
}

async function createIndex(txn: TransactionExecutor): Promise<void> {
    await txn.execute("CREATE INDEX ON People (firstName)");
}

async function insertDocument(txn: TransactionExecutor): Promise<void> {
    const person: Record<string, any> = {
        firstName: "John",
        lastName: "Doe",
        age: 42
    };
    await txn.execute("INSERT INTO People ?", person);
}

async function fetchDocuments(txn: TransactionExecutor): Promise<dom.Value[]> {
    return (await txn.execute("SELECT firstName, age, lastName FROM People WHERE
 firstName = ?", "John")).getResultList();
}

async function updateDocuments(txn: TransactionExecutor): Promise<void> {
    await txn.execute("UPDATE People SET lastName = ? WHERE firstName = ?",
 "Stiles", "John");
};

async function main(): Promise<void> {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");

    const resultList: dom.Value[] = await driver.executeLambda(async (txn:
 TransactionExecutor) => {
        console.log("Create table People");
        await createTable(txn);
        console.log("Create index on firstName");
        await createIndex(txn);
        console.log("Insert document");
        await insertDocument(txn);
        console.log("Fetch document");
        await fetchDocuments(txn);
        console.log("Update document");
        await updateDocuments(txn);
        console.log("Fetch document after update");
        return await fetchDocuments(txn);
    });
```

```
    // Pretty print the result list
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
}

if (require.main === module) {
    main();
}
```

Geben Sie den folgenden Befehl ein, um die vollständige Anwendung auszuführen.

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

Amazon QLDB-Treiber für Node.js — Kochbuch-Referenz

Dieses Referenzhandbuch zeigt häufige Anwendungsfälle des Amazon QLDB-Treibers für Node.js. Sie bietet eine Reihe von TypeScript Codebeispielen für die Erstellungs JavaScript -, Lese-, Lese-, Lese-, Lese-, Lese-, Aktualisierungs-, Löschvorgänge (CRUD). Es enthält auch Codebeispiele für die Verarbeitung von Amazon Ion-Daten. Darüber hinaus werden in diesem Leitfaden bewährte Verfahren vorgestellt, um Transaktionen idempotent zu machen und Eindeutigkeitsbeschränkungen zu implementieren.

Inhalt

Importieren des Treibers

Das folgende Codebeispiel importiert den Treiber.

JavaScript

```
var qldb = require('amazon-qldb-driver-nodejs');
var ionjs = require('ion-js');
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { dom, dumpBinary, load } from "ion-js";
```

> **Note**
>
> In diesem Beispiel wird auch das Amazon Ion-Paket (`ion-js`) importiert. Sie benötigen dieses Paket, um Ion-Daten zu verarbeiten, wenn Sie einige Datenoperationen in dieser Referenz ausführen. Weitere Informationen hierzu finden Sie unter Arbeiten mit Amazon Ion-on.

Instantiieren des Treibers

Das folgende Codebeispiel erstellt eine Instanz des Treibers, die mithilfe der Standardeinstellungen eine Verbindung zu einem angegebenen Ledger-Namen herstellt.

JavaScript

```
const qldbDriver = new qldb.QldbDriver("vehicle-registration");
```

TypeScript

```
const qldbDriver: QldbDriver = new QldbDriver("vehicle-registration");
```

CRUD-Operationen

QLDB führt die CRUD-Operationen (Erstellen, Lesen, Aktualisieren, Löschen) als Teil einer Transaktion durch.

> **Warning**
>
> Als bewährte Methode sollten Sie Ihre Schreibtransaktionen strikt idempotent gestalten.

Transaktionen idempotent machen

Wir empfehlen, Schreibtransaktionen idempotent zu machen, um unerwartete Nebenwirkungen bei Wiederholungsversuchen zu vermeiden. Eine Transaktion ist idempotent, wenn sie mehrfach ausgeführt werden kann und jedes Mal zu identischen Ergebnissen führt.

Stellen Sie sich zum Beispiel eine Transaktion vor, bei der ein Dokument in eine Tabelle mit dem Namen eingefügt wird`Person`. Die Transaktion sollte zunächst prüfen, ob das Dokument bereits in der Tabelle vorhanden ist. Ohne diese Überprüfung enthält die Tabelle möglicherweise doppelte Dokumente.

Nehmen wir an, QLDB führt die Transaktion auf der Serverseite erfolgreich durch, aber der Client läuft beim Warten auf eine Antwort ein Timeout. Wenn die Transaktion nicht idempotent ist, kann dasselbe Dokument im Falle eines erneuten Versuchs mehrmals eingefügt werden.

Verwendung von Indizes zur Vermeidung vollständiger Tabellenscans

Wir empfehlen außerdem, Anweisungen mit einer`WHERE` Prädikatklausel auszuführen, indem Sie einen Gleichheitsoperator für ein indiziertes Feld oder eine Dokument-ID verwenden, z. B.`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`. Ohne diese indizierte Suche muss QLDB einen Tabellenscan durchführen, was zu Transaktions-Timeouts oder OCC-Konflikten (Optimist Concurrency Control) führen kann.

Weitere Informationen zu OCC finden Sie unter Amazon QLDB-Nebenläufigkeit von Amazon QLDB-Nebenläufigkeit.

Implizit erstellte Transaktionen

Die Methode QldbDriver.executeLambda akzeptiert eine Lambda-Funktion, die eine Instanz von empfängt TransactionExecutor, mit der Sie Anweisungen ausführen können. Die Instanz von`TransactionExecutor` umschließt eine implizit erstellte Transaktion.

Sie können Anweisungen innerhalb der Lambda-Funktion ausführen, indem Sie die Execute-Methode des Transaction Executors verwenden. Der Treiber schreibt die Transaktion implizit fest, wenn die Lambda-Funktion zurückkehrt.

> **Note**
>
> Die`execute` Methode unterstützt sowohl Amazon Ion-Typen als auch native Typen von Node.js. Wenn Sie einen systemeigenen Typ von Node.js als Argument an übergeben`execute`, konvertiert der Treiber ihn mithilfe des`ion-js` Pakets in einen Ion-Typ (vorausgesetzt, dass die Konvertierung für den angegebenen Datentyp Node.js unterstützt wird). Unterstützte Datentypen und Konvertierungsregeln finden Sie in der Ion JavaScript DOM README-Datei.

In den folgenden Abschnitten wird gezeigt, wie grundlegende CRUD-Operationen ausgeführt, eine benutzerdefinierte Wiederholungslogik angegeben und Eindeutigkeitsbeschränkungen implementiert werden.

Inhalt

- Einfügen von Dokumenten
  - Einfügen mehrerer Dokumente in eine Anweisung
- Dokumente aktualisieren
- Dokumente löschen
- Ausführen mehrerer Kontoauszüge in einer Transaktion
- Wiederholen Sie dieses Logiken für
- Implementieren von Einheitseinschränkungen für Richtlinien

Erstellen von Tabellen

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        await txn.execute("CREATE TABLE Person");
    });
})();
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        await txn.execute('CREATE TABLE Person');
    });
}());
```

Erstellen von Indizes

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        await txn.execute("CREATE INDEX ON Person (GovId)");
    });
})();
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        await txn.execute('CREATE INDEX ON Person (GovId)');
    });
}());
```

Dokumente lesen

JavaScript

```
(async function() {
    // Assumes that Person table has documents as follows:
    // { "GovId": "TOYENC486FH", "FirstName": "Brent" }
    await qldbDriver.executeLambda(async (txn) => {
        const results = (await txn.execute("SELECT * FROM Person WHERE GovId =
 'TOYENC486FH'")).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    // Assumes that Person table has documents as follows:
    // { "GovId": "TOYENC486FH", "FirstName": "Brent" }
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute("SELECT * FROM Person WHERE
 GovId = 'TOYENC486FH'")).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

Verwenden von Abfrageparametern

Das folgende Codebeispiel verwendet einen Abfrageparameter vom systemeigenen Typ.

JavaScript

```
(async function() {
    // Assumes that Person table has documents as follows:
    // { "GovId": "TOYENC486FH", "FirstName": "Brent" }
    await qldbDriver.executeLambda(async (txn) => {
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 'TOYENC486FH')).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    // Assumes that Person table has documents as follows:
    // { "GovId": "TOYENC486FH", "FirstName": "Brent" }
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', 'TOYENC486FH')).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

Das folgende Codebeispiel verwendet einen Abfrageparameter vom Typ Ion.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const govId = ionjs.load("TOYENC486FH");
```

```
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 govId)).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const govId: dom.Value = load("TOYENC486FH");

        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', govId)).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

Das folgende Codebeispiel verwendet mehrere Abfrageparameter.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ? AND
 FirstName = ?', 'TOYENC486FH', 'Brent')).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
```

```
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ? AND FirstName = ?', 'TOYENC486FH', 'Brent')).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

Das folgende Codebeispiel verwendet eine Liste von Abfrageparametern.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const govIds = ['TOYENC486FH','LOGANB486CG','LEWISR261LL'];
        /*
        Assumes that Person table has documents as follows:
        { "GovId": "TOYENC486FH", "FirstName": "Brent" }
        { "GovId": "LOGANB486CG", "FirstName": "Brent" }
        { "GovId": "LEWISR261LL", "FirstName": "Raul" }
        */
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId IN
 (?,?,?)', ...govIds)).getResultList();
        for (let result of results) {
            console.log(result.get('GovId'));
            console.log(result.get('FirstName'));
            /*
            prints:
            [String: 'TOYENC486FH']
            [String: 'Brent']
            [String: 'LOGANB486CG']
            [String: 'Brent']
            [String: 'LEWISR261LL']
            [String: 'Raul']
            */
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const govIds: string[] = ['TOYENC486FH','LOGANB486CG','LEWISR261LL'];
        /*
        Assumes that Person table has documents as follows:
        { "GovId": "TOYENC486FH", "FirstName": "Brent" }
        { "GovId": "LOGANB486CG", "FirstName": "Brent" }
        { "GovId": "LEWISR261LL", "FirstName": "Raul" }
        */
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId IN (?,?,?)', ...govIds)).getResultList();
        for (let result of results) {
            console.log(result.get('GovId'));
            console.log(result.get('FirstName'));
            /*
            prints:
            [String: 'TOYENC486FH']
            [String: 'Brent']
            [String: 'LOGANB486CG']
            [String: 'Brent']
            [String: 'LEWISR261LL']
            [String: 'Raul']
            */
        }
    });
}());
```

> **Note**
>
> Wenn Sie eine Abfrage ohne eine indizierte Suche ausführen, wird ein vollständiger Tabellenscan aufgerufen. In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Abfragen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Einfügen von Dokumenten

Das folgende Codebeispiel fügt native Datentypen ein.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        // Check if doc with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 'TOYENC486FH')).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            const doc = {
                'FirstName': 'Brent',
                'GovId': 'TOYENC486FH',
            };
            await txn.execute('INSERT INTO Person ?', doc);
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        // Check if doc with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', 'TOYENC486FH')).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            const doc: Record<string, string> = {
                'FirstName': 'Brent',
                'GovId': 'TOYENC486FH',
            };
            await txn.execute('INSERT INTO Person ?', doc);
        }
    });
}());
```

Das folgende Codebeispiel fügt Ion-Datentypen ein.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        // Check if doc with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 'TOYENC486FH')).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            const doc = {
                'FirstName': 'Brent',
                'GovId': 'TOYENC486FH',
            };
            // Create a sample Ion doc
            const ionDoc = ionjs.load(ionjs.dumpBinary(doc));

            await txn.execute('INSERT INTO Person ?', ionDoc);
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        // Check if doc with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', 'TOYENC486FH')).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            const doc: Record<string, string> = {
                'FirstName': 'Brent',
                'GovId': 'TOYENC486FH',
            };
            // Create a sample Ion doc
            const ionDoc: dom.Value = load(dumpBinary(doc));

            await txn.execute('INSERT INTO Person ?', ionDoc);
        }
    });
}());
```

Diese Transaktion fügt ein Dokument in die`Person` Tabelle ein. Vor dem Einfügen wird zunächst geprüft, ob das Dokument bereits in der Tabelle vorhanden ist. Diese Prüfung macht die Transaktion idempotenter Natur. Selbst wenn Sie diese Transaktion mehrmals ausführen, verursacht sie keine unbeabsichtigten Nebenwirkungen.

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Einfügen mehrerer Dokumente in eine Anweisung

Um mehrere Dokumente mit einer einzigenINSERT Anweisung einzufügen, können Sie wie folgt einen Parameter vom Typ list an die Anweisung übergeben.

```
// people is a list
txn.execute("INSERT INTO People ?", people);
```

Sie schließen den variablen Platzhalter (?) nicht in doppelte eckige Klammern (`<<...>>`) ein, wenn Sie eine Liste übergeben. In manuellen PartiQL-Anweisungen bezeichnen doppelte spitze Klammern eine ungeordnete Sammlung, die als Bag bezeichnet wird.

Dokumente aktualisieren

Das folgende Codebeispiel verwendet native Datentypen.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        await txn.execute('UPDATE Person SET FirstName = ? WHERE GovId = ?', 'John',
 'TOYENC486FH');
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
```

```
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        await txn.execute('UPDATE Person SET FirstName = ? WHERE GovId = ?', 'John',
 'TOYENC486FH');
    });
}());
```

Das folgende Codebeispiel verwendet Ion-Datentypen.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const firstName = ionjs.load("John");
        const govId = ionjs.load("TOYENC486FH");

        await txn.execute('UPDATE Person SET FirstName = ? WHERE GovId = ?',
 firstName, govId);
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const firstName: dom.Value = load("John");
        const govId: dom.Value = load("TOYENC486FH");

        await txn.execute('UPDATE Person SET FirstName = ? WHERE GovId = ?',
 firstName, govId);
    });
}());
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Dokumente löschen

Das folgende Codebeispiel verwendet native Datentypen.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        await txn.execute('DELETE FROM Person WHERE GovId = ?', 'TOYENC486FH');
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        await txn.execute('DELETE FROM Person WHERE GovId = ?', 'TOYENC486FH');
    });
}());
```

Das folgende Codebeispiel verwendet Ion-Datentypen.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const govId = ionjs.load("TOYENC486FH");

        await txn.execute('DELETE FROM Person WHERE GovId = ?', govId);
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const govId: dom.Value = load("TOYENC486FH");

        await txn.execute('DELETE FROM Person WHERE GovId = ?', govId);
    });
}());
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Ausführen mehrerer Kontoauszüge in einer Transaktion

TypeScript

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
async function insureCar(driver: QldbDriver, vin: string): Promise<boolean> {

    return await driver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
 vin)).getResultList();

        if (results.length > 0) {
            await txn.execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin);
            return true;
        }
        return false;
    });
};
```

Wiederholen Sie dieses Logiken für

Die`executeLambda` Methode des Treibers verfügt über einen integrierten Wiederholungsmechanismus, der die Transaktion erneut versucht, wenn eine wiederholbare Ausnahme auftritt (z. B. Timeouts oder OCC-Konflikte). Die maximale Anzahl an Wiederholungen und die Backoff-Strategie sind konfigurierbar.

Das Standardlimit für Wiederholungsversuche ist4, und die Standard-Backoff-Strategie basiert auf [defaultBackoffFunction](#)einer Basis von`10` Millisekunden. Sie können die Wiederholungskonfiguration

pro Treiberinstanz und auch pro Transaktion festlegen, indem Sie eine Instanz von verwenden RetryConfig.

Das folgende Codebeispiel spezifiziert die Wiederholungslogik mit einem benutzerdefinierten Wiederholungslimit und einer benutzerdefinierten Backoff-Strategie für eine Instanz des Treibers.

JavaScript

```
var qldb = require('amazon-qldb-driver-nodejs');

// Configuring retry limit to 2
const retryConfig = new qldb.RetryConfig(2);
const qldbDriver = new qldb.QldbDriver("test-ledger", undefined, undefined,
 retryConfig);

// Configuring a custom backoff which increases delay by 1s for each attempt.
const customBackoff = (retryAttempt, error, transactionId) => {
    return 1000 * retryAttempt;
};

const retryConfigCustomBackoff = new qldb.RetryConfig(2, customBackoff);
const qldbDriverCustomBackoff = new qldb.QldbDriver("test-ledger", undefined,
 undefined, retryConfigCustomBackoff);
```

TypeScript

```
import { BackoffFunction, QldbDriver, RetryConfig } from "amazon-qldb-driver-nodejs"

// Configuring retry limit to 2
const retryConfig: RetryConfig = new RetryConfig(2);
const qldbDriver: QldbDriver = new QldbDriver("test-ledger", undefined, undefined,
 retryConfig);

// Configuring a custom backoff which increases delay by 1s for each attempt.
const customBackoff: BackoffFunction = (retryAttempt: number, error: Error,
 transactionId: string) => {
    return 1000 * retryAttempt;
};

const retryConfigCustomBackoff: RetryConfig = new RetryConfig(2, customBackoff);
const qldbDriverCustomBackoff: QldbDriver = new QldbDriver("test-ledger", undefined,
 undefined, retryConfigCustomBackoff);
```

Das folgende Codebeispiel spezifiziert die Wiederholungslogik mit einem benutzerdefinierten Wiederholungslimit und einer benutzerdefinierten Backoff-Strategie für eine bestimmte Lambda-Ausführung. Diese Konfiguration für`executeLambda` überschreibt die Wiederholungslogik, die für die Treiberinstanz festgelegt ist.

JavaScript

```
var qldb = require('amazon-qldb-driver-nodejs');

// Configuring retry limit to 2
const retryConfig1 = new qldb.RetryConfig(2);
const qldbDriver = new qldb.QldbDriver("test-ledger", undefined, undefined,
 retryConfig1);

// Configuring a custom backoff which increases delay by 1s for each attempt.
const customBackoff = (retryAttempt, error, transactionId) => {
    return 1000 * retryAttempt;
};

const retryConfig2 = new qldb.RetryConfig(2, customBackoff);

// The config `retryConfig1` will be overridden by `retryConfig2`
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        await txn.execute('CREATE TABLE Person');
    }, retryConfig2);
}());
```

TypeScript

```
import { BackoffFunction, QldbDriver, RetryConfig, TransactionExecutor } from
 "amazon-qldb-driver-nodejs"

// Configuring retry limit to 2
const retryConfig1: RetryConfig = new RetryConfig(2);
const qldbDriver: QldbDriver = new QldbDriver("test-ledger", undefined, undefined,
 retryConfig1);

// Configuring a custom backoff which increases delay by 1s for each attempt.
const customBackoff: BackoffFunction = (retryAttempt: number, error: Error,
 transactionId: string) => {
    return 1000 * retryAttempt;
};
```

```
const retryConfig2: RetryConfig = new RetryConfig(2, customBackoff);

// The config `retryConfig1` will be overridden by `retryConfig2`
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        await txn.execute('CREATE TABLE Person');
    }, retryConfig2);
}());
```

Implementieren von Einheitseinschränkungen für Richtlinien

QLDB unterstützt keine eindeutigen Indizes, aber Sie können dieses Verhalten in Ihrer Anwendung implementieren.

Angenommen, Sie möchten eine Eindeutigkeitsbeschränkung für das`GovId` Feld in der`Person` Tabelle implementieren. Dazu können Sie eine Transaktion schreiben, die Folgendes bewirkt:

1. Stellen Sie sicher, dass die Tabelle keine vorhandenen Dokumente mit einem bestimmten Wert enthält`GovId`.
2. Fügen Sie das Dokument ein, wenn die Assertion erfolgreich ist.

Besteht eine konkurrierende Transaktion gleichzeitig die Assertion, wird nur eine der Transaktionen erfolgreich abgeschlossen. Die andere Transaktion schlägt mit einer OCC-Konfliktexception fehl.

Das folgende Codebeispiel veranschaulicht Sie, wie Sie, wie Sie diese Einheitseinschränkungslogik implementieren.

JavaScript

```
const govId = 'TOYENC486FH';
const document = {
    'FirstName': 'Brent',
    'GovId': 'TOYENC486FH',
};
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        // Check if doc with GovId = govId exists
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 govId)).getResultList();
        // Insert the document after ensuring it doesn't already exist
```

```
        if (results.length == 0) {
            await txn.execute('INSERT INTO Person ?', document);
        }
    });
})();
```

TypeScript

```
const govId: string = 'TOYENC486FH';
const document: Record<string, string> = {
    'FirstName': 'Brent',
    'GovId': 'TOYENC486FH',
};
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        // Check if doc with GovId = govId exists
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', govId)).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            await txn.execute('INSERT INTO Person ?', document);
        }
    });
})();
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Arbeiten mit Amazon Ion-on

In den folgenden Abschnitten sehen Sie, wie Sie, wie Sie, wie Sie Ion-Daten verarbeiten.

Inhalt

- Einen Ion-Textdump abrufen

Import des Ionen-Moduls

JavaScript

```
var ionjs = require('ion-js');
```

TypeScript

```
import { dom, dumpBinary, dumpText, load } from "ion-js";
```

Erstellen von Ion-Typen

Das folgende Codebeispiel erstellt ein Ion-Objekt aus Ion-Text.

JavaScript

```
const ionText  = '{GovId: "TOYENC486FH", FirstName: "Brent"}';
const ionObj = ionjs.load(ionText);

console.log(ionObj.get('GovId')); // prints [String: 'TOYENC486FH']
console.log(ionObj.get('FirstName')); // prints [String: 'Brent']
```

TypeScript

```
const ionText: string = '{GovId: "TOYENC486FH", FirstName: "Brent"}';
const ionObj: dom.Value = load(ionText);

console.log(ionObj.get('GovId')); // prints [String: 'TOYENC486FH']
console.log(ionObj.get('FirstName')); // prints [String: 'Brent']
```

Das folgende Codebeispiel erstellt ein Ion-Objekt aus einem Node.js -Wörterbuch.

JavaScript

```
const aDict = {
    'GovId': 'TOYENC486FH',
    'FirstName': 'Brent'
```

```
};
const ionObj = ionjs.load(ionjs.dumpBinary(aDict));
console.log(ionObj.get('GovId')); // prints [String: 'TOYENC486FH']
console.log(ionObj.get('FirstName')); // prints [String: 'Brent']
```

TypeScript

```
const aDict: Record<string, string> = {
    'GovId': 'TOYENC486FH',
    'FirstName': 'Brent'
};
const ionObj: dom.Value = load(dumpBinary(aDict));
console.log(ionObj.get('GovId')); // prints [String: 'TOYENC486FH']
console.log(ionObj.get('FirstName')); // prints [String: 'Brent']
```

Einen Ion-Binärdump abrufen

JavaScript

```
// ionObj is an Ion struct
console.log(ionjs.dumpBinary(ionObj).toString()); // prints
 224,1,0,234,238,151,129,131,222,147,135,190,144,133,71,111,118,73,100,137,70,105,114,115,11
```

TypeScript

```
// ionObj is an Ion struct
console.log(dumpBinary(ionObj).toString()); // prints
 224,1,0,234,238,151,129,131,222,147,135,190,144,133,71,111,118,73,100,137,70,105,114,115,11
```

Einen Ion-Textdump abrufen

JavaScript

```
// ionObj is an Ion struct
console.log(ionjs.dumpText(ionObj)); // prints
 {GovId:"TOYENC486FH",FirstName:"Brent"}
```

TypeScript

```
// ionObj is an Ion struct
```

```
console.log(dumpText(ionObj)); // prints {GovId:"TOYENC486FH",FirstName:"Brent"}
```

Weitere Informationen zu Ion finden Sie in der Amazon Ion-Dokumentation unter GitHub. Weitere Codebeispiele für die Arbeit mit Ion in QLDB finden Sie unterArbeiten mit Amazon Ion-Datentypen in Amazon QLDB.

Amazon QLDB-Treiber für Python

Um mit Daten in Ihrem Ledger zu arbeiten, können Sie von Ihrer Python-Anwendung aus mithilfe einesAWS bereitgestellten Treibers eine Verbindung zu Amazon QLDB herstellen. In den folgenden Themen wird die ersten Schritte durchführen: den QLDB-Treiber für Python tun.

Themen

- Ressourcen für Fahrer
- Voraussetzungen
- Installation
- Amazon QLDB-Treiber für Python — Schnellstart-Tutorial
- Amazon QLDB-Treiber für Python — Kochbuch-Referenz

Ressourcen für Fahrer

Weitere Informationen zu den vom Python-Treiber unterstützten Funktionen finden Sie in den folgenden Ressourcen:

- API-Referenz: 3.x, 2.x
- Treiber-Quellcode (GitHub)
- Quellcode der Beispielanwendung (GitHub)
- Beispiele für Amazon Ion-Code

Voraussetzungen

Bevor Sie mit dem QLDB-Treiber für Python beginnen, müssen Sie die folgenden Schritte durchführen:

1. Befolgen Sie die AWS-Einrichtungsanweisungen unter Zugreifen auf Amazon QLDB. Diese umfasst die folgenden Funktionen:

   1. Registrieren Sie sich beiAWS.
   2. Erstellen Sie einen Benutzer mit den entsprechenden QLDB-Berechtigungen.
   3. Erteilen programmgesteuerten Zugriffs für die Entwicklung
2. Installieren Sie eine der folgenden Versionen von Python von der Python-Downloadseite:

   - 3.6 oder später — QLDB-Treiber für Python v3
   - 3.4 oder später — QLDB-Treiber für Python v2
3. Richten Sie IhreAWS Anmeldeinformationen und Ihre Standardeinstellungen einAWS-Region. Weitere Informationen finden Sie unter Schnellstart in der AWS SDK for Python (Boto3)-Dokumentation.

   Eine vollständige Liste der verfügbaren Regionen finden Sie unter Amazon QLDB-Endpunkte und Kontingente in der Allgemeine AWS-Referenz.

Als Nächstes können Sie die vollständige Beispielanwendung für das Tutorial herunterladen — oder Sie können nur den Treiber in einem Python-Projekt installieren und kurze Codebeispiele ausführen.

- Um den QLDB-Treiber und denAWS SDK for Python (Boto3) in einem vorhandenen Projekt zu installieren, fahren Sie mit fortInstallation.
- Informationen zum Einrichten eines Projekts und zur Ausführung von kurzen Codebeispielen, die grundlegende Datentransaktionen in einem Ledger veranschaulichen, finden Sie unterErste Schritte.
- Ausführlichere Beispiele für Daten- und Verwaltungs-API-Operationen finden Sie in der vollständigen Beispielanwendung des Tutorials unterPython-Tutorial.

Installation

QLDB unterstützt die folgenden Treiberversionen und ihre Python-Abhängigkeiten.

Treiberversion	Python-Version	Status	Datum der letzten Veröffentlichung
2.x	3.4 oder später	Produktionsfreigabe	7. Mai 2020

Treiberversion	Python-Version	Status	Datum der letzten Veröffentlichung
3.x	3.6 oder höher	Produktionsfreigabe	28. Oktober 2021

Um den QLDB-Treiber von PyPI aus mit`pip` (einem Paketmanager für Python) zu installieren, geben Sie in der Befehlszeile Folgendes ein.

3.x

```
pip install pyqldb
```

2.x

```
pip install pyqldb==2.0.2
```

Durch die Installation des Treibers werden auch seine Abhängigkeiten installiert, einschließlich der Pakete AWS SDK for Python (Boto3)und Amazon Ion.

Verwenden des Treibers zum Herstellen einer Verbindung mit einem Ledger

Dann können Sie den Treiber importieren und mit ihm eine Verbindung zu einem Ledger herstellen. Im folgenden Python-Codebeispiel wird gezeigt, wie eine Sitzung für einen angegebenen Ledger-Namen erstellt wird.

3.x

```
from pyqldb.driver.qldb_driver import QldbDriver
qldb_driver = QldbDriver(ledger_name='testLedger')

for table in qldb_driver.list_tables():
    print(table)
```

2.x

```
from pyqldb.driver.pooled_qldb_driver import PooledQldbDriver

qldb_driver = PooledQldbDriver(ledger_name='testLedger')
```

```
qldb_session = qldb_driver.get_session()

for table in qldb_session.list_tables():
    print(table)
```

Kurze Codebeispiele für die Ausführung grundlegender Datentransaktionen in einem Ledger finden Sie unterKochbuchreferenz.

Amazon QLDB-Treiber für Python — Schnellstart-Tutorial

In diesem Tutorial erfahren Sie, wie Sie eine einfache Anwendung mit der neuesten Version des Amazon QLDB-Treibers für Python einrichten. Dieses Handbuch enthält Schritte zum Installieren des Treibers und kurze Codebeispiele für grundlegende CRUD-Vorgänge (Create, Read, Update und Delete). Ausführlichere Beispiele, die diese Vorgänge in einer vollständigen Beispielanwendung veranschaulichen, finden Sie im Python-Tutorial.

Themen

- Voraussetzungen
- Schritt 1: Einrichten des Projekts
- Schritt 2: Initialisieren des Treibers
- Schritt 3: Erstellen einer Tabelle und eines Index
- Schritt 4: Einfügen eines Dokuments
- Schritt 5: Abfragen des Dokuments
- Schritt 6: Aktualisieren des Dokuments
- Ausführen der vollständigen Anwendung

Voraussetzungen

Bevor Sie beginnen, stellen Sie sicher, dass die folgende Voraussetzung erfüllt ist:

1. Füllen Sie dasVoraussetzungen für den Python-Treiber aus, falls noch nicht geschehen, um das für den Python-Treiber zu tun. Dazu gehören die RegistrierungAWS, die Gewährung von programmatischen Zugriff für die Entwicklung und die Installation von Python-Version 3.6 oder höher.
2. Ledger mit dem Namen `quick-start` erstellen.

Informationen zum Erstellen eines Ledgers finden Sie unterGrundfunktionen für Amazon QLDB-Ledgers oderSchritt 1: Erstellen eines neuen Ledgers unter Erste Schritte mit der Konsole.

Schritt 1: Einrichten des Projekts

Erste Schritte

> **Note**
>
> Wenn Sie eine IDE verwenden, die Funktionen zur Automatisierung dieser Einrichtungsschritte enthält, können Sie mit dem Vorgang fortfahrenSchritt 2: Initialisieren des Treibers.

1. Erstellen Sie einen Ordner für Ihre Anwendung.

   ```
   $ mkdir myproject
   $ cd myproject
   ```

2. Um den QLDB-Treiber für Python von PyPI zu installieren, geben Sie den folgendenpip Befehl ein.

   ```
   $ pip install pyqldb
   ```

   Durch die Installation des Treibers werden auch seine Abhängigkeiten installiert, einschließlich der Pakete AWS SDK for Python (Boto3)und Amazon Ion.

3. Erstellen Sie eine neue Datei mit dem Namen `app.py`.

   Fügen Sie dann schrittweise die Codebeispiele in den folgenden Schritten hinzu, um einige grundlegende CRUD-Operationen auszuprobieren. Oder Sie können das step-by-step Tutorial überspringen und stattdessen die komplette Anwendung ausführen.

Schritt 2: Initialisieren des Treibers

Initialisieren Sie eine Instance des Treibers, der eine Verbindung mit dem Ledger `quick-start` herstellt. Fügen Sie nun folgenden Code in die Datei `app.py` ein:

```
from pyqldb.config.retry_config import RetryConfig
from pyqldb.driver.qldb_driver import QldbDriver

# Configure retry limit to 3
retry_config = RetryConfig(retry_limit=3)

# Initialize the driver
print("Initializing the driver")
qldb_driver = QldbDriver("quick-start", retry_config=retry_config)
```

Schritt 3: Erstellen einer Tabelle und eines Index

Im folgenden Codebeispiel wird veranschaulicht, wie `CREATE TABLE`- und `CREATE INDEX`-Anweisungen ausgeführt werden.

Fügen Sie den folgenden Code hinzu, der eine Tabelle mit dem Namen `People` und einen Index für das Feld `lastName` in dieser Tabelle erstellt. Indizes sind erforderlich, um die Abfrageleistung zu optimieren und dabei zu helfen, Konfliktausnahmen für Optimist Concurrency Control (OCC) zu begrenzen.

```
def create_table(transaction_executor):
    print("Creating a table")
    transaction_executor.execute_statement("Create TABLE People")

def create_index(transaction_executor):
    print("Creating an index")
    transaction_executor.execute_statement("CREATE INDEX ON People(lastName)")

# Create a table
qldb_driver.execute_lambda(lambda executor: create_table(executor))

# Create an index on the table
qldb_driver.execute_lambda(lambda executor: create_index(executor))
```

Schritt 4: Einfügen eines Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine `INSERT`-Anweisung ausgeführt wird. QLDB unterstützt die PartiQL-Abfragesprache (SQL-kompatibel) und das Amazon Ion-Datenformat (Superset von JSON).

Fügen Sie den folgenden Code hinzu, der ein Dokument in die Tabelle `People` einfügt.

```
def insert_documents(transaction_executor, arg_1):
    print("Inserting a document")
    transaction_executor.execute_statement("INSERT INTO People ?", arg_1)

# Insert a document
doc_1 = { 'firstName': "John",
          'lastName': "Doe",
          'age': 32,
        }

qldb_driver.execute_lambda(lambda x: insert_documents(x, doc_1))
```

In diesem Beispiel wird ein Fragezeichen (?) als Variablenplatzhalter verwendet, um die Dokumentinformationen an die Anweisung zu übergeben. Die`execute_statement` Methode unterstützt Werte sowohl in Amazon Ion-Typen als auch in systemeigenen Python-Typen.

> **Tip**
>
> Um mehrere Dokumente mit einer einzigenINSERT Anweisung einzufügen, können Sie wie folgt einen Parameter vom Typ list an die Anweisung übergeben.
>
> ```
> # people is a list
> transaction_executor.execute_statement("INSERT INTO Person ?", people)
> ```
>
> Sie schließen den variablen Platzhalter (?) nicht in doppelte eckige Klammern (`<<...>>`) ein, wenn Sie eine Liste übergeben. In manuellen PartiQL-Anweisungen bezeichnen doppelte spitze Klammern eine ungeordnete Sammlung, die als Bag bezeichnet wird.

Schritt 5: Abfragen des Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine `SELECT`-Anweisung ausgeführt wird.

Fügen Sie den folgenden Code hinzu, der ein Dokument aus der Tabelle `People` abfragt.

```
def read_documents(transaction_executor):
    print("Querying the table")
    cursor = transaction_executor.execute_statement("SELECT * FROM People WHERE
 lastName = ?", 'Doe')
```

```
    for doc in cursor:
        print(doc["firstName"])
        print(doc["lastName"])
        print(doc["age"])

# Query the table
qldb_driver.execute_lambda(lambda executor: read_documents(executor))
```

Schritt 6: Aktualisieren des Dokuments

Im folgenden Codebeispiel wird veranschaulicht, wie eine UPDATE-Anweisung ausgeführt wird.

1. Fügen Sie den folgenden Code hinzu, der ein Dokument in der`People` Tabelle aktualisiert, in`age` es auf aktualisiert wird`42`.

   ```
   def update_documents(transaction_executor, age, lastName):
       print("Updating the document")
       transaction_executor.execute_statement("UPDATE People SET age = ? WHERE
    lastName = ?", age, lastName)

   # Update the document
   age = 42
   lastName = 'Doe'

   qldb_driver.execute_lambda(lambda x: update_documents(x, age, lastName))
   ```

2. Fragen Sie die Tabelle erneut ab, um den aktualisierten Wert zu sehen.

   ```
   # Query the updated document
   qldb_driver.execute_lambda(lambda executor: read_documents(executor))
   ```

3. Verwenden Sie den folgenden, um die Anwendung auszuführen, um die Anwendung auszuführen, um die Anwendung auszuführen, um die Anwendung auszuführen.

   ```
   $ python app.py
   ```

Ausführen der vollständigen Anwendung

Das folgende Codebeispiel ist die vollständige Version der`app.py` Anwendung. Anstatt die vorherigen Schritte einzeln auszuführen, können Sie dieses Codebeispiel auch kopieren und von

Anfang bis Ende ausführen. Diese Anwendung demonstriert einige grundlegende CRUD-Operationen für den Ledger namens `quick-start`.

> **Note**
>
> Bevor Sie diesen Code ausführen, stellen Sie sicher, dass Sie noch keine aktive Tabelle mit dem Namen `People` im `quick-start`-Ledger besitzen.

```
from pyqldb.config.retry_config import RetryConfig
from pyqldb.driver.qldb_driver import QldbDriver

def create_table(transaction_executor):
    print("Creating a table")
    transaction_executor.execute_statement("CREATE TABLE People")

def create_index(transaction_executor):
    print("Creating an index")
    transaction_executor.execute_statement("CREATE INDEX ON People(lastName)")

def insert_documents(transaction_executor, arg_1):
    print("Inserting a document")
    transaction_executor.execute_statement("INSERT INTO People ?", arg_1)

def read_documents(transaction_executor):
    print("Querying the table")
    cursor = transaction_executor.execute_statement("SELECT * FROM People WHERE
 lastName = ?", 'Doe')


    for doc in cursor:
        print(doc["firstName"])
        print(doc["lastName"])
        print(doc["age"])

def update_documents(transaction_executor, age, lastName):
    print("Updating the document")
    transaction_executor.execute_statement("UPDATE People SET age = ? WHERE lastName
 = ?", age, lastName)

# Configure retry limit to 3
retry_config = RetryConfig(retry_limit=3)
```

```
# Initialize the driver
print("Initializing the driver")
qldb_driver = QldbDriver("quick-start", retry_config=retry_config)

# Create a table
qldb_driver.execute_lambda(lambda executor: create_table(executor))

# Create an index on the table
qldb_driver.execute_lambda(lambda executor: create_index(executor))

# Insert a document
doc_1 = { 'firstName': "John",
          'lastName': "Doe",
          'age': 32,
        }

qldb_driver.execute_lambda(lambda x: insert_documents(x, doc_1))

# Query the table
qldb_driver.execute_lambda(lambda executor: read_documents(executor))

# Update the document
age = 42
lastName = 'Doe'

qldb_driver.execute_lambda(lambda x: update_documents(x, age, lastName))

# Query the table for the updated document
qldb_driver.execute_lambda(lambda executor: read_documents(executor))
```

Verwenden Sie den folgenden Befehl aus Ihrem Projektverzeichnis, um die vollständige Anwendung auszuführen, um die vollständige Anwendung auszuführen, um die vollständige Anwendung auszuführen.

```
$ python app.py
```

Amazon QLDB-Treiber für Python — Kochbuch-Referenz

Dieses Referenzhandbuch zeigt häufige Anwendungsfälle des Amazon QLDB-Treibers für Python. Es enthält Pyon-Codebeispiele, die zeigen, wie CRUD-Operationen (Erstellen, Lesen, Aktualisieren, Löschen) ausführen. Es enthält auch Codebeispiele für die Verarbeitung von Amazon Ion-Daten.

Darüber hinaus werden in diesem Leitfaden bewährte Verfahren vorgestellt, um Transaktionen idempotent zu machen und Eindeutigkeitsbeschränkungen zu implementieren.

> **Note**
>
> Gegebenenfalls haben einige Anwendungsfälle unterschiedliche Codebeispiele für jede unterstützte Hauptversion des QLDB-Treibers für Python.

Inhalt

Importieren des Treibers

Das folgende Codebeispiel importiert den Treiber.

3.x

```
from pyqldb.driver.qldb_driver import QldbDriver
import amazon.ion.simpleion as simpleion
```

2.x

```
from pyqldb.driver.pooled_qldb_driver import PooledQldbDriver
import amazon.ion.simpleion as simpleion
```

> **Note**
>
> In diesem Beispiel wird auch das Amazon Ion-Paket (`amazon.ion.simpleion`) importiert. Sie benötigen dieses Paket, um Ion-Daten zu verarbeiten, wenn Sie einige Datenoperationen in dieser Referenz ausführen. Weitere Informationen hierzu finden Sie unter Arbeiten mit Amazon Ion-Technologie.

instanziieren des Treibers

Das folgende Codebeispiel erstellt eine Instanz des Treibers, die mithilfe der Standardeinstellungen eine Verbindung zu einem angegebenen Ledger-Namen herstellt.

3.x

```
qldb_driver = QldbDriver(ledger_name='vehicle-registration')
```

2.x

```
qldb_driver = PooledQldbDriver(ledger_name='vehicle-registration')
```

CRUD-Operationen

QLDB führt CRUD-Operationen (Erstellen, Lesen, Aktualisieren, Löschen) als Teil einer Transaktion durch.

> **Warning**
>
> Als bewährte Methode sollten Schreibtransaktionen strikt idempotent sein.

Transaktionen idempotent machen

Wir empfehlen, Schreibtransaktionen idempotent zu machen, um unerwartete Nebenwirkungen bei Wiederholungsversuchen zu vermeiden. Eine Transaktion ist idempotent, wenn sie mehrfach ausgeführt werden kann und jedes Mal zu identischen Ergebnissen führt.

Stellen Sie sich zum Beispiel eine Transaktion vor, bei der ein Dokument in eine Tabelle mit dem Namen eingefügt wird`Person`. Die Transaktion sollte zunächst prüfen, ob das Dokument bereits in der Tabelle vorhanden ist. Ohne diese Überprüfung enthält die Tabelle möglicherweise doppelte Dokumente.

Nehmen wir an, QLDB führt die Transaktion auf der Serverseite erfolgreich durch, aber der Client läuft beim Warten auf eine Antwort ein Timeout. Wenn die Transaktion nicht idempotent ist, kann dasselbe Dokument im Falle eines erneuten Versuchs mehrmals eingefügt werden.

Verwendung von Indizes zur Vermeidung vollständiger Tabellenscans

Wir empfehlen außerdem, Anweisungen mit einer`WHERE` Prädikatklausel auszuführen, indem Sie einen Gleichheitsoperator für ein indiziertes Feld oder eine Dokument-ID verwenden, z. B.`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`. Ohne diese indizierte Suche muss QLDB einen Tabellenscan durchführen, was zu Transaktions-Timeouts oder OCC-Konflikten (Optimist Concurrency Control) führen kann.

Weitere Informationen zu OCC finden Sie unter Amazon QLDB-Nebenläufigkeit von Amazon QLDB-Nebenläufigkeit.

Implizit erstellte Transaktionen

Die Methode pyqldb.driver.qldb_driver.execute_lambda akzeptiert eine Lambda-Funktion, die eine Instanz von pyqldb.execution.executor.Executor empfängt, mit der Sie Anweisungen ausführen können. Die Instanz von`Executor` umschließt eine implizit erstellte Transaktion.

Sie können Anweisungen innerhalb der Lambda-Funktion ausführen, indem Sie die Methode execute_statement des Transaktionsausführers verwenden. Der Treiber schreibt die Transaktion implizit fest, wenn die Lambda-Funktion zurückkehrt.

> **Note**
>
> Die`execute_statement` Methode unterstützt sowohl Amazon Ion-Typen als auch native Python-Typen. Wenn Sie einen nativen Python-Typ als Argument an übergeben`execute_statement`, konvertiert der Treiber ihn mithilfe des`amazon.ion.simpleion` Moduls in einen Ion-Typ (vorausgesetzt, dass die Konvertierung für den angegebenen Python-Datentyp unterstützt wird). Unterstützte Datentypen und Konvertierungsregeln finden Sie im Simpleion-Quellcode.

In den folgenden Abschnitten wird gezeigt, wie grundlegende CRUD-Operationen ausgeführt, eine benutzerdefinierte Wiederholungslogik angegeben und Eindeutigkeitsbeschränkungen implementiert werden.

Inhalt

Erstellen von Tabellen

```
def create_table(transaction_executor):
    transaction_executor.execute_statement("CREATE TABLE Person")

qldb_driver.execute_lambda(lambda executor: create_table(executor))
```

Erstellen von Indizes

```
def create_index(transaction_executor):
    transaction_executor.execute_statement("CREATE INDEX ON Person(GovId)")

qldb_driver.execute_lambda(lambda executor: create_index(executor))
```

Dokumente lesen

```
# Assumes that Person table has documents as follows:
# { "GovId": "TOYENC486FH", "FirstName": "Brent" }

def read_documents(transaction_executor):
    cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId =
 'TOYENC486FH'")

    for doc in cursor:
        print(doc["GovId"]) # prints TOYENC486FH
        print(doc["FirstName"]) # prints Brent

qldb_driver.execute_lambda(lambda executor: read_documents(executor))
```

Verwenden von Abfrageparametern

Das folgende Codebeispiel verwendet einen Abfrageparameter vom systemeigenen Typ.

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?",
 'TOYENC486FH')
```

Das folgende Codebeispiel verwendet einen Abfrageparameter vom Typ Ion.

```
name = ion.loads('Brent')
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE FirstName
 = ?", name)
```

Das folgende Codebeispiel verwendet mehrere Abfrageparameter.

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?
 AND FirstName = ?", 'TOYENC486FH', "Brent")
```

Das folgende Codebeispiel verwendet eine Liste von Abfrageparametern.

```
gov_ids = ['TOYENC486FH','ROEE1','YH844']
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId IN
 (?,?,?)", *gov_ids)
```

> **Note**
>
> Wenn Sie eine Abfrage ohne eine indizierte Suche ausführen, wird ein vollständiger Tabellenscan aufgerufen. In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Abfragen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Einfügen von Dokumenten

Das folgende Codebeispiel fügt native Datentypen ein.

```
def insert_documents(transaction_executor, arg_1):
    # Check if doc with GovId:TOYENC486FH exists
    # This is critical to make this transaction idempotent
    cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId
 = ?", 'TOYENC486FH')
    # Check if there is any record in the cursor
    first_record = next(cursor, None)

    if first_record:
        # Record already exists, no need to insert
        pass
    else:
        transaction_executor.execute_statement("INSERT INTO Person ?", arg_1)

doc_1 = { 'FirstName': "Brent",
          'GovId': 'TOYENC486FH',
        }

qldb_driver.execute_lambda(lambda executor: insert_documents(executor, doc_1))
```

Das folgende Codebeispiel fügt Ion-Datentypen ein.

```
def insert_documents(transaction_executor, arg_1):
```

```
    # Check if doc with GovId:TOYENC486FH exists
    # This is critical to make this transaction idempotent
    cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId
 = ?", 'TOYENC486FH')
    # Check if there is any record in the cursor
    first_record = next(cursor, None)

    if first_record:
        # Record already exists, no need to insert
        pass
    else:
        transaction_executor.execute_statement("INSERT INTO Person ?", arg_1)

doc_1 = { 'FirstName': 'Brent',
          'GovId': 'TOYENC486FH',
        }

# create a sample Ion doc
ion_doc_1 = simpleion.loads(simpleion.dumps(doc_1))

qldb_driver.execute_lambda(lambda executor: insert_documents(executor, ion_doc_1))
```

Diese Transaktion fügt ein Dokument in die`Person` Tabelle ein. Vor dem Einfügen wird zunächst geprüft, ob das Dokument bereits in der Tabelle vorhanden ist. Diese Prüfung macht die Transaktion idempotenter Natur. Selbst wenn Sie diese Transaktion mehrmals ausführen, verursacht sie keine unbeabsichtigten Nebenwirkungen.

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Einfügen mehrerer Dokumente in eine Anweisung

Um mehrere Dokumente mit einer einzigenINSERT Anweisung einzufügen, können Sie wie folgt einen Parameter vom Typ list an die Anweisung übergeben.

```
# people is a list
transaction_executor.execute_statement("INSERT INTO Person ?", people)
```

Sie schließen den variablen Platzhalter (?) nicht in doppelte eckige Klammern (<<...>>) ein, wenn Sie eine Liste übergeben. In manuellen PartiQL-Anweisungen bezeichnen doppelte spitze Klammern eine ungeordnete Sammlung, die als Bag bezeichnet wird.

Dokumente aktualisieren

Das folgende Codebeispiel verwendet native Datentypen.

```
def update_documents(transaction_executor, gov_id, name):
    transaction_executor.execute_statement("UPDATE Person SET FirstName = ?  WHERE
 GovId = ?", name, gov_id)

gov_id = 'TOYENC486FH'
name = 'John'

qldb_driver.execute_lambda(lambda executor: update_documents(executor, gov_id, name))
```

Das folgende Codebeispiel verwendet Ion-Datentypen.

```
def update_documents(transaction_executor, gov_id, name):
    transaction_executor.execute_statement("UPDATE Person SET FirstName = ? WHERE GovId
 = ?", name, gov_id)

# Ion datatypes
gov_id = simpleion.loads('TOYENC486FH')
name = simpleion.loads('John')

qldb_driver.execute_lambda(lambda executor: update_documents(executor, gov_id, name))
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Dokumente löschen

Das folgende Codebeispiel verwendet native Datentypen.

```
def delete_documents(transaction_executor, gov_id):
```

```
    cursor = transaction_executor.execute_statement("DELETE FROM Person WHERE GovId
 = ?", gov_id)

gov_id = 'TOYENC486FH'

qldb_driver.execute_lambda(lambda executor: delete_documents(executor, gov_id))
```

Das folgende Codebeispiel verwendet Ion-Datentypen.

```
def delete_documents(transaction_executor, gov_id):
    cursor = transaction_executor.execute_statement("DELETE FROM Person WHERE GovId
 = ?", gov_id)

# Ion datatypes
gov_id = simpleion.loads('TOYENC486FH')

qldb_driver.execute_lambda(lambda executor: delete_documents(executor, gov_id))
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Ausführen mehrerer Kontoauszüge in einer Transaktion

```
# This code snippet is intentionally trivial. In reality you wouldn't do this because
 you'd
# set your UPDATE to filter on vin and insured, and check if you updated something or
 not.

def do_insure_car(transaction_executor, vin):
    cursor = transaction_executor.execute_statement(
        "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE", vin)
    first_record = next(cursor, None)
    if first_record:
        transaction_executor.execute_statement(
            "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin)
        return True
    else:
        return False
```

```
def insure_car(qldb_driver, vin_to_insure):
    return qldb_driver.execute_lambda(
        lambda executor: do_insure_car(executor, vin_to_insure))
```

Wiederholungslogik

Die`execute_lambda` Methode des Treibers verfügt über einen integrierten Wiederholungsmechanismus, der die Transaktion erneut versucht, wenn eine wiederholbare Ausnahme auftritt (z. B. Timeouts oder OCC-Konflikte).

3.x

Die maximale Anzahl an Wiederholungsversuchen und die Backoff-Strategie sind konfigurierbar.

Das Standardlimit für Wiederholungsversuche ist`4`, und die Standard-Backoff-Strategie ist Exponentieller Backoff und Jitter mit einer Basis von`10` Millisekunden. Sie können die Wiederholungskonfiguration pro Treiberinstanz und auch pro Transaktion festlegen, indem Sie eine Instanz von pyqldb.config.retry_config verwenden. RetryConfig.

Das folgende Codebeispiel spezifiziert die Wiederholungslogik mit einem benutzerdefinierten Wiederholungslimit und einer benutzerdefinierten Backoff-Strategie für eine Instanz des Treibers.

```
from pyqldb.config.retry_config import RetryConfig
from pyqldb.driver.qldb_driver import QldbDriver

# Configuring retry limit to 2
retry_config = RetryConfig(retry_limit=2)
qldb_driver = QldbDriver("test-ledger", retry_config=retry_config)

# Configuring a custom backoff which increases delay by 1s for each attempt.
def custom_backoff(retry_attempt, error, transaction_id):
    return 1000 * retry_attempt

retry_config_custom_backoff = RetryConfig(retry_limit=2,
 custom_backoff=custom_backoff)
qldb_driver = QldbDriver("test-ledger", retry_config=retry_config_custom_backoff)
```

Das folgende Codebeispiel spezifiziert die Wiederholungslogik mit einem benutzerdefinierten Wiederholungslimit und einer benutzerdefinierten Backoff-Strategie für eine bestimmte Lambda-Ausführung. Diese Konfiguration für`execute_lambda` überschreibt die Wiederholungslogik, die für die Treiberinstanz festgelegt ist.

```
from pyqldb.config.retry_config import RetryConfig
from pyqldb.driver.qldb_driver import QldbDriver

# Configuring retry limit to 2
retry_config_1 = RetryConfig(retry_limit=4)
qldb_driver = QldbDriver("test-ledger", retry_config=retry_config_1)

# Configuring a custom backoff which increases delay by 1s for each attempt.
def custom_backoff(retry_attempt, error, transaction_id):
    return 1000 * retry_attempt

retry_config_2 = RetryConfig(retry_limit=2, custom_backoff=custom_backoff)

# The config `retry_config_1` will be overriden by `retry_config_2`
qldb_driver.execute_lambda(lambda txn: txn.execute_statement("CREATE TABLE Person"),
 retry_config_2)
```

2.x

Die maximale Anzahl an Wiederholungsversuchen ist konfigurierbar. Sie können das Wiederholungslimit konfigurieren, indem Sie die`retry_limit` Eigenschaft bei der Initialisierung festlegen`PooledQldbDriver`.

Das Standardlimit für Wiederholungsversuche ist4.

Implementierung von Eindeutigkeitseinschränkungen

QLDB unterstützt keine eindeutigen Indizes, aber Sie können dieses Verhalten in Ihrer Anwendung implementieren.

Angenommen, Sie möchten eine Eindeutigkeitsbeschränkung für das`GovId` Feld in der`Person` Tabelle implementieren. Dazu können Sie eine Transaktion schreiben, die Folgendes bewirkt:

1. Stellen Sie sicher, dass die Tabelle keine vorhandenen Dokumente mit einem bestimmten Wert enthält`GovId`.
2. Fügen Sie das Dokument ein, wenn die Assertion erfolgreich ist.

Besteht eine konkurrierende Transaktion gleichzeitig die Assertion, wird nur eine der Transaktionen erfolgreich abgeschlossen. Die andere Transaktion schlägt mit einer OCC-Konfliktexception fehl.

Das folgende Codebeispiel veranschaulicht, wie diese Logik für Eindeutigkeitseinschränkungen implementiert wird.

```
def insert_documents(transaction_executor, gov_id, document):
    # Check if doc with GovId = gov_id exists
    cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId
 = ?", gov_id)
    # Check if there is any record in the cursor
    first_record = next(cursor, None)

    if first_record:
        # Record already exists, no need to insert
        pass
    else:
        transaction_executor.execute_statement("INSERT INTO Person ?", document)

qldb_driver.execute_lambda(lambda executor: insert_documents(executor, gov_id,
 document))
```

> **Note**
>
> In diesem Beispiel empfehlen wir, einen Index für das`GovId` Feld zu haben, um die Leistung zu optimieren. Ohne aktiviertes`GovId` Index können Anweisungen mehr Latenz haben und auch zu OCC-Konfliktausnahmen oder Transaktions-Timeouts führen.

Arbeiten mit Amazon Ion-Technologie

In den folgenden Abschnitten sehen Sie, wie das Amazon-Ion-Modul für die Verarbeitung von Ion-Daten verwendet werden.

Inhalt

Import des Ionen-Moduls

```
import amazon.ion.simpleion as simpleion
```

Erstellen von Ion-Typen

Das folgende Codebeispiel erstellt ein Ion-Objekt aus Ion-Text.

```
ion_text = '{GovId: "TOYENC486FH", FirstName: "Brent"}'
ion_obj = simpleion.loads(ion_text)

print(ion_obj['GovId']) # prints TOYENC486FH
print(ion_obj['Name']) # prints Brent
```

Das folgende Codebeispiel erstellt ein Ion-Objekt aus einem Python`dict`.

```
a_dict = { 'GovId': 'TOYENC486FH',
           'FirstName': "Brent"
         }
ion_obj = simpleion.loads(simpleion.dumps(a_dict))

print(ion_obj['GovId']) # prints TOYENC486FH
print(ion_obj['FirstName']) # prints Brent
```

Einen Ion-Binärdump abrufen

```
# ion_obj is an Ion struct
print(simpleion.dumps(ion_obj)) # b'\xe0\x01\x00\xea\xee\x97\x81\x83\xde\x93\x87\xbe
\x90\x85GovId\x89FirstName\xde\x94\x8a\x8bTOYENC486FH\x8b\x85Brent'
```

Einen Ion-Textdump abrufen

```
# ion_obj is an Ion struct
print(simpleion.dumps(ion_obj, binary=False)) # prints $ion_1_0
 {GovId:'TOYENC486FH',FirstName:"Brent"}
```

Weitere Informationen zur Arbeit mit Ion finden Sie in der Amazon Ion-Dokumentation unter GitHub. Weitere Codebeispiele für die Arbeit mit Ion in QLDB finden Sie unterArbeiten mit Amazon Ion-Datentypen in Amazon QLDB.

Grundlegendes zur Sitzungsverwaltung mit dem Treiber in Amazon QLDB

Wenn Sie Erfahrung mit einem relationalen Datenbankmanagementsystem (RDBMS) haben, sind Sie möglicherweise mit gleichzeitigen Verbindungen vertraut. QLDB hat nicht das gleiche Konzept wie eine herkömmliche RDBMS-Verbindung, da Transaktionen mit HTTP-Anfrage- und Antwortnachrichten ausgeführt werden.

In QLDB ist das analoge Konzept eine aktive Sitzung. Eine Sitzung ähnelt konzeptionell einer Benutzeranmeldung — sie verwaltet Informationen über Ihre Datentransaktionsanforderungen an ein Ledger. Eine aktive Sitzung ist eine Sitzung, in der aktiv eine Transaktion ausgeführt wird. Es kann sich auch um eine Sitzung handeln, die kürzlich eine Transaktion abgeschlossen hat und bei der der Dienst davon ausgeht, dass er sofort eine weitere Transaktion starten wird. QLDB unterstützt eine aktiv laufende Transaktion pro Sitzung.

Das Limit für gleichzeitige aktive Sitzungen pro Ledger ist in Kontingente und Limits in Amazon QLDB definiert. Wenn dieses Limit erreicht ist, führt jede Sitzung, die versucht, eine Transaktion zu starten, zu einem Fehler (`LimitExceededException`).

Bewährte Methoden für die Konfiguration eines Sitzungspool in Ihrer Anwendung mithilfe des QLDB-Treibers finden SieKonfigurieren von QldbDriver Objekt in den Amazon QLDB-Treiberempfehlungen.

Inhalt

-Lebenszyklus von Sitzungen

Die folgende Abfolge von QLDB-Sitzungs-API-Vorgängen stellt den typischen Lebenszyklus einer QLDB-Sitzung dar:

1. `StartSession`

2. `StartTransaction`
3. `ExecuteStatement`
4. `CommitTransaction`
5. Wiederholen Sie die Schritte 2—4, um mehr Transaktionen zu starten (jeweils eine Transaktion).
6. `EndSession`

Sitzungsablauf

QLDB läuft ab und verwirft eine Sitzung nach einer Gesamtlebensdauer von 13—17 Minuten, unabhängig davon, ob aktiv eine Transaktion ausgeführt wird. Sitzungen können aus einer Reihe von Gründen verloren gehen oder beeinträchtigt werden, wie z. B. Hardwarefehler, Netzwerkfehler oder Anwendungsneustarts. QLDB erzwingt eine maximale Lebensdauer von Sitzungen, um sicherzustellen, dass die Client-Anwendung gegen Sitzungsausfälle resistent ist.

Sitzungsverarbeitung im QLDB-Treiber

Sie können zwar einAWS SDK verwenden, um direkt mit der QLDB-Sitzungs-API zu interagieren, dies erhöht jedoch die Komplexität und erfordert die Berechnung eines Commit-Digest. QLDB verwendet diesen Commit-Digest, um die Integrität der Transaktion sicherzustellen. Anstatt direkt mit dieser API zu interagieren, empfehlen wir die Verwendung des QLDB-Treibers.

Der Treiber bietet eine abstrakte Ebene auf hoher Ebene über der Transaktionsdaten-API. Es optimiert den Prozess der Ausführung von PartiQL-Anweisungen auf Ledger-Daten durch die Verwaltung von SendCommandAPI-Aufrufen. Für diese API-Aufrufe sind mehrere Parameter erforderlich, die der Treiber für Sie verarbeitet, darunter die Verwaltung von Sitzungen, Transaktionen und Richtlinien für Wiederholungen bei Fehlern.

Themen

- Übersicht über das Sitzungspoling
- Sitzungspoling und Transaktionslogik
- Wiederaufnahme der Sessions an den Pool

Übersicht über das Sitzungspoling

In älteren Versionen des QLDB-Treibers (z. B. Java v1.1.0) haben wir zwei Implementierungen des Treiberobjekts bereitgestellt: einen Standard, Nonpooled`QldbDriver` und einePooled`QldbDriver`.

Wie der Name schon sagt,`PooledQldbDriver` verwaltet der einen Pool von Sitzungen, die transaktionsübergreifend wiederverwendet werden.

Basierend auf dem Feedback der Benutzer ziehen es Entwickler vor, die Pooling-Funktion und ihre Vorteile standardmäßig zu nutzen, anstatt den Standardtreiber zu verwenden. Also haben wir die entferntPooledQldbDriver und die Session-Pooling-Funktion auf verschobenQldbDriver. Diese Änderung ist in der neuesten Version jedes Treibers enthalten (z. B. Java v2.0.0).

Der Treiber bietet drei Abstraktionsebenen:

- Treiber (gepoolte Treiberimplementierung) — Die Abstraktion auf oberster Ebene. Der Treiber verwaltet und verwaltet einen Pool von Sitzungen. Wenn Sie den Treiber bitten, eine Transaktion auszuführen, wählt der Treiber eine Sitzung aus dem Pool aus und verwendet sie, um die Transaktion auszuführen. Wenn die Transaktion aufgrund eines Sitzungsfehlers (`InvalidSessionException`) fehlschlägt, verwendet der Treiber eine andere Sitzung, um die Transaktion erneut zu versuchen. Im Wesentlichen bietet der Treiber ein vollständig verwaltetes Sitzungserlebnis.
- Sitzung — Eine Ebene unter der Treiberabstraktion. Eine Sitzung ist in einem Pool enthalten, und der Treiber verwaltet den Lebenszyklus der Sitzung. Wenn eine Transaktion fehlschlägt, versucht der Treiber sie bis zu einer bestimmten Anzahl von Versuchen mit derselben Sitzung erneut. Wenn die Sitzung jedoch auf einen Fehler (`InvalidSessionException`) stößt, verwirft QLDB ihn intern. Der Treiber ist dann dafür verantwortlich, eine weitere Sitzung aus dem Pool abzurufen, um die Transaktion erneut zu versuchen.
- Transaktion — Die niedrigste Abstraktionsebene. Eine Transaktion ist in einer Sitzung enthalten, und die Sitzung verwaltet den Lebenszyklus der Transaktion. Die Sitzung ist dafür verantwortlich, die Transaktion im Fehlerfall erneut zu versuchen. Die Sitzung stellt außerdem sicher, dass keine offene Transaktion durchsickert, die nicht bestätigt oder abgebrochen wurde.

In der neuesten Version jedes Treibers können Sie Operationen nur auf der Treiberabstraktionsebene ausführen. Sie haben keine direkte Kontrolle über einzelne Sitzungen und Transaktionen (das heißt, es gibt keine API-Operationen zum manuellen Starten einer neuen Sitzung oder Transaktion).

Sitzungspoling und Transaktionslogik

Die neueste Version jedes Treibers bietet keine gepoolte Implementierung des Treiberobjekts mehr. Standardmäßig verwaltet das`QldbDriver` Objekt den Sitzungspool. Wenn Sie einen Anruf tätigen, um eine Transaktion auszuführen, führt der Treiber die folgenden Schritte aus:

1. Der Treiber überprüft, ob er das Limit für den Sitzungspool erreicht hat. Wenn ja, löst der Fahrer sofort eine`NoSessionAvailable` Ausnahme aus. Andernfalls fährt sie mit dem nächsten Schritt fort.
2. Der Treiber prüft, ob der Pool über eine verfügbare Sitzung verfügt.
   - Wenn eine Sitzung im Pool verfügbar ist, verwendet der Treiber sie, um eine Transaktion auszuführen.
   - Wenn im Pool keine Sitzung verfügbar ist, erstellt der Treiber eine neue Sitzung und verwendet sie, um eine Transaktion auszuführen.
3. Nachdem der Treiber in Schritt 2 eine Sitzung erhalten hat, ruft der Treiber den`execute` Vorgang auf der Sitzungsinstanz auf.
4. Während der`execute` Sitzung versucht der Treiber, eine Transaktion durch einen Aufruf zu starten`startTransaction`.
   - Wenn die Sitzung nicht gültig ist, schlägt der`startTransaction` Anruf fehl und der Treiber kehrt zu Schritt 1 zurück.
   - Wenn der`startTransaction` Anruf erfolgreich ist, fährt der Fahrer mit dem nächsten Schritt fort.
5. Der Treiber führt Ihren Lambda-Ausdruck aus. Dieser Lambda-Ausdruck kann einen oder mehrere Aufrufe zur Ausführung von PartiQL-Anweisungen enthalten. Nachdem der Lambda-Ausdruck ohne Fehler ausgeführt wurde, fährt der Treiber fort, die Transaktion zu bestätigen.
6. Der Commit der Transaktion kann zu einem von zwei Ergebnissen führen:
   - Der Commit ist erfolgreich und der Treiber gibt die Kontrolle an Ihren Anwendungscode zurück.
   - Das Commit schlägt aufgrund eines optimistischen Concurrency Control (OCC) -Konflikts fehl. In diesem Fall wiederholt der Treiber die Schritte 4—6 in derselben Sitzung. Die maximale Anzahl an Wiederholungen ist in Ihrem Anwendungscode konfigurierbar. Das Standardlimit ist4.

> **Note**
>
> Wenn in den Schritten 4—6 ein ausgelöst`InvalidSessionException` wird, markiert der Treiber die Sitzung als geschlossen und kehrt zu Schritt 1 zurück, um die Transaktion erneut zu versuchen.
> Wenn in den Schritten 4—6 eine weitere Ausnahme ausgelöst wird, prüft der Treiber, ob die Ausnahme erneut versucht werden kann. Wenn ja, wird die Transaktion bis zur angegebenen

Anzahl von Wiederholungsversuchen wiederholt. Wenn nicht, wird die Ausnahme auf Ihren Anwendungscode übertragen (explodiert und ausgelöst).

Wiederaufnahme der Sessions an den Pool

`InvalidSessionException`Tritt während einer aktiven Transaktion zu irgendeinem Zeitpunkt eine auf, gibt der Treiber die Sitzung nicht an den Pool zurück. QLDB verwirft stattdessen die Sitzung, und der Treiber erhält eine weitere Sitzung aus dem Pool. In allen anderen Fällen gibt der Treiber die Sitzung an den Pool zurück.

Amazon QLDB-Treiberempfehlungen

In diesem Abschnitt werden bewährte Methoden zum Konfigurieren und Verwenden des Amazon QLDB-Treibers für jede unterstützte Sprache beschrieben. Bei den Codebeispielen handelt es sich um Java-Beispiele.

Diese Empfehlungen gelten für die meisten typischen Anwendungsfälle, aber nicht für alle. Verwenden Sie die folgenden Empfehlungen, wenn Sie glauben, dass sie für Ihre Anwendung geeignet sind.

Themen

Konfigurieren von QldbDriver Objekt

Die`QldbDriver`object verwaltet Verbindungen zu Ihrem Ledger, indem ein Pool vonSitzungendie über Transaktionen hinweg wiederverwendet werden. EINSitzungstellt eine einzelne Verbindung zum Ledger dar. QLDB unterstützt eine aktiv laufende Transaktion pro Sitzung.

> **Important**
>
> Bei älteren Treiberversionen befindet sich die Funktion zum Session-Pooling immer noch im`PooledQldbDriver`Objekt anstelle von`QldbDriver`aus. Wenn

Sie eine der folgenden Versionen verwenden, ersetzen Sie alle Erwähnungen von`QldbDriver`mit`PooledQldbDriver`für den Rest dieses Themas.

Treiber	Version
Java	`1.1.0`oder früher
.NET	`0.1.0-beta`
Node.js	`1.0.0-rc.1` oder früher
Python	`2.0.2`oder früher

Die`PooledQldbDriver`-Objekt ist in der neuesten Version der Treiber veraltet. Es wird empfohlen, ein Upgrade auf die neueste Version durchzuführen und alle Instanzen von`PooledQldbDriver`zu`QldbDriver`aus.

Konfiguration QldbDriver als globales Objekt

Um die Verwendung von Treibern und Sitzungen zu optimieren, müssen Sie sicherstellen, dass nur eine globale Instance des Treibers in Ihrer Anwendungs-Instance vorhanden ist. In Java können Sie beispielsweise Abhängigkeits-Injection-Frameworks wie Spring, Google Guice oder Dagger verwenden. Das folgende Codebeispiel zeigt, wie Sie `QldbDriver` als Singleton konfigurieren.

```
@Singleton
public QldbDriver qldbDriver (AWSCredentialsProvider credentialsProvider,
                                    @Named(LEDGER_NAME_CONFIG_PARAM) String ledgerName)
 {
    QldbSessionClientBuilder builder = QldbSessionClient.builder();
    if (null != credentialsProvider) {
        builder.credentialsProvider(credentialsProvider);
    }
    return QldbDriver.builder()
            .ledger(ledgerName)
            .transactionRetryPolicy(RetryPolicy
                .builder()
                .maxRetries(3)
                .build())
            .sessionClientBuilder(builder)
```

```
            .build();
}
```

Konfigurieren der Wiederholungsversuche

Der Treiber wiederholt Transaktionen automatisch, wenn häufige vorübergehende Ausnahmen (z. B.`SocketTimeoutException`oder`NoHttpResponseException`) treten auf. Um die maximale Anzahl an Wiederholungsversuchen festzulegen, können Sie das`maxRetries`-Parameter des`transactionRetryPolicy`Konfigurationsobjekt beim Erstellen einer Instanz von`QldbDriver`aus. (Verwenden Sie für ältere Treiberversionen, wie im vorherigen Abschnitt aufgeführt,`retryLimit`-Parameter von`PooledQldbDriver`.)

Der Standardwert von `maxRetries` ist `4`.

Bei clientseitigen Fehlern wie `InvalidParameterException` sind keine wiederholten Versuche möglich. Wenn sie auftreten, wird die Transaktion abgebrochen, die Sitzung wird an den Pool zurückgegeben, und eine Ausnahme wird für den Client des Treibers ausgelöst.

Konfigurieren der maximalen Anzahl gleichzeitiger Sitzungen und Transaktionen

Die maximale Anzahl an Ledger-Sitzungen, die von einer Instance von`QldbDriver`Transaktionen auszuführen, wird durch seine`maxConcurrentTransactions`-Parameter. (Für ältere Treiberversionen, wie im vorherigen Abschnitt aufgeführt, wird dies durch die`poolLimit`-Parameter von`PooledQldbDriver`.)

Dieser Grenzwert muss größer als Null und kleiner oder gleich der maximalen Anzahl offener HTTP-Verbindungen sein, die der Sitzungs-Client zulässt (definiert nach Maßgabe des jeweiligen AWS-SDKS. In Java wird die maximale Anzahl an Verbindungen beispielsweise im Objekt ClientConfiguration festgelegt.

Der Standardwert von`maxConcurrentTransactions`ist die maximale Verbindungseinstellung Ihres AWS-SDKS.

Wenn Sie `QldbDriver` in der Anwendung konfigurieren, beachten Sie die folgenden Aspekte hinsichtlich der Skalierung:

- Der Pool sollte immer mindestens so viele Sitzungen enthalten, wie gleichzeitig ausgeführte Transaktionen geplant sind.
- In einem Multi-Thread-Modell, in dem ein Supervisor-Thread an Worker-Threads delegiert, sollte der Treiber mindestens so viele Sitzungen enthalten, wie Worker-Threads vorliegen. Andernfalls warten Threads bei Spitzenlast nacheinander auf eine verfügbare Sitzung.

- Das Service-Limit für gleichzeitige aktive Sitzungen pro Ledger ist in Kontingente und Limits in Amazon QLDB definiert. Stellen Sie sicher, dass Sie nicht mehr als dieses Limit an gleichzeitigen Sitzungen für einen einzelnen Ledger über alle Clients konfigurieren.

Erneuter Versuch bei Ausnahmen

Beachten Sie beim erneuten Versuch bei Ausnahmen, die in QLDB auftreten, die folgenden Empfehlungen.

Wiederholen von OccConflictException

Optimistic Concurrency Control(OCC) Konflikte treten auf, wenn sich die Daten, auf die Transaktion zugreift, seit dem Beginn der Transaktion geändert haben. QLDB löst diese Ausnahme aus, während sie versucht, die Transaktion zu übernehmen. Der Fahrer versucht die Transaktion bis zu so oft wie`maxRetries`ist konfiguriert.

Weitere Informationen zu OCC und Best Practices zur Verwendung von Indizes zur Begrenzung von OCC-Konflikten finden Sie unterAmazon QLDB-Nebenläufigkeit von Amazon QLDB-Nebenläufigkeitaus.

Andere Ausnahmen außerhalb von QlDbDriver erneut versuchen

Um eine Transaktion außerhalb des Treibers zu wiederholen, wenn benutzerdefinierte, anwendungsdefinierte Ausnahmen während der Laufzeit ausgelöst werden, müssen Sie die Transaktion umbrechen. Der folgende Code zeigt beispielsweise, wie Sie dasReslience4J-Bibliothek, um eine Transaktion in QLDB erneut zu versuchen.

```
private final RetryConfig retryConfig = RetryConfig.custom()
        .maxAttempts(MAX_RETRIES)
        .intervalFunction(IntervalFunction.ofExponentialRandomBackoff())
        // Retry this exception
        .retryExceptions(InvalidSessionException.class, MyRetryableException.class)
        // But fail for any other type of exception extended from RuntimeException
        .ignoreExceptions(RuntimeException.class)
        .build();

// Method callable by a client
public void myTransactionWithRetries(Params params) {
    Retry retry = Retry.of("registerDriver", retryConfig);

    Function<Params, Void> transactionFunction = Retry.decorateFunction(
```

```
            retry,
            parameters ->  transactionNoReturn(params));
    transactionFunction.apply(params);
}

private Void transactionNoReturn(Params params) {
    try (driver.execute(txn -> {
            // Transaction code
        });
    }
    return null;
}
```

> **Note**
>
> Das erneute Versuch einer Transaktion außerhalb des QLDB-Treibers hat einen Multiplikatoreffekt. Zum Beispiel, wenn`QldbDriver`ist so konfiguriert, dass drei Wiederholungen versucht werden, und die benutzerdefinierte Wiederholungslogik wiederholt sich ebenfalls drei Wiederholungen, dieselbe Transaktion kann bis zu neun Mal wiederholt werden.

Transaktionen idempotent machen

Als bewährte Methode sollten Sie Ihre Schreibtransaktionen idempotent machen, um unerwartete Nebenwirkungen im Falle von Wiederholungen zu vermeiden. Eine Transaktion istidempotentwenn es mehrmals laufen kann und jedes Mal identische Ergebnisse erzielen kann.

Weitere Informationen hierzu finden Sie unter Amazon QLDB-Nebenläufigkeit von Amazon QLDB-Nebenläufigkeit.

Optimierung der Leistung

Um die Leistung beim Ausführen von Transaktionen mit dem Treiber zu optimieren, sollten Sie Folgendes berücksichtigen:

- Die`execute`Betrieb macht immer mindestens drei`SendCommand`-Aufrufe der API an QLDB, die folgende Befehle enthalten:
  1. `StartTransaction`
  2. `ExecuteStatement`

Dieser Befehl wird für jede PartiQL-Anweisung aufgerufen, die Sie im`execute`blockieren.

3. `CommitTransaction`

Berücksichtigen Sie die Gesamtzahl der auszuführenden API-Aufrufe, wenn Sie die Gesamt-Workload der Anwendung kalkulieren.

- Grundsätzlich empfehlen wir, mit einem Single-Thread-Writer zu beginnen und Transaktionen zu optimieren, indem mehrere Anweisungen in einer Transaktion zusammengefasst werden. Maximieren Sie die Kontingente für Transaktionsgröße, Dokumentgröße und Anzahl der Dokumente pro Transaktion gemäß der Definition in Kontingente und Limits in Amazon QLDB.
- Wenn das Batching für große Transaktionslasten nicht ausreicht, können Sie es mit Multi-Threading versuchen, indem Sie weitere Writer hinzufügen. Sie sollten jedoch sorgfältig die Anwendungsanforderungen an die Dokument- und Transaktionssequenzierung im Hinblick auf die damit einhergehende Komplexität abwägen.

Ausführen mehrerer Anweisungen pro Transaktion

Wie in dervorheriger Abschnitt, können Sie mehrere Anweisungen pro Transaktion ausführen, um die Leistung Ihrer Anwendung zu optimieren. Im folgenden Codebeispiel fragen Sie eine Tabelle ab und aktualisieren dann ein Dokument in dieser Tabelle innerhalb einer Transaktion. Hierzu übergeben Sie einen Lambda-Ausdruck an`execute`verwenden.

Java

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
public static boolean InsureCar(QldbDriver qldbDriver, final String vin) {
    final IonSystem ionSystem = IonSystemBuilder.standard().build();
    final IonString ionVin = ionSystem.newString(vin);

    return qldbDriver.execute(txn -> {
        Result result = txn.execute(
                "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
                ionVin);
        if (!result.isEmpty()) {
            txn.execute("UPDATE Vehicles SET insured = TRUE WHERE vin = ?", ionVin);
            return true;
```

```
        }
        return false;
    });
}
```

.NET

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
public static async Task<bool> InsureVehicle(IAsyncQldbDriver driver, string vin)
{
    ValueFactory valueFactory = new ValueFactory();
    IIonValue ionVin = valueFactory.NewString(vin);

    return await driver.Execute(async txn =>
    {
        // Check if the vehicle is insured.
        Amazon.QLDB.Driver.IAsyncResult result = await txn.Execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
 ionVin);

        if (await result.CountAsync() > 0)
        {
            // If the vehicle is not insured, insure it.
            await txn.Execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", ionVin);
            return true;
        }
        return false;
    });
}
```

Go

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
func InsureCar(driver *qldbdriver.QLDBDriver, vin string) (bool, error) {
    insured, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
```

```
        result, err := txn.Execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE", vin)
        if err != nil {
            return false, err
        }

        hasNext := result.Next(txn)
        if !hasNext && result.Err() != nil {
            return false, result.Err()
        }

        if hasNext {
            _, err = txn.Execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin)
            if err != nil {
                return false, err
            }
            return true, nil
        }
        return false, nil
    })
    if err != nil {
        panic(err)
    }

    return insured.(bool), err
}
```

Node.js

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
async function insureCar(driver: QldbDriver, vin: string): Promise<boolean> {

    return await driver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
 vin)).getResultList();

        if (results.length > 0) {
```

```
            await txn.execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin);
            return true;
        }
        return false;
    });
};
```

Python

```
# This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
# set your UPDATE to filter on vin and insured, and check if you updated something
 or not.

def do_insure_car(transaction_executor, vin):
    cursor = transaction_executor.execute_statement(
        "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE", vin)
    first_record = next(cursor, None)
    if first_record:
        transaction_executor.execute_statement(
            "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin)
        return True
    else:
        return False

def insure_car(qldb_driver, vin_to_insure):
    return qldb_driver.execute_lambda(
        lambda executor: do_insure_car(executor, vin_to_insure))
```

Die des Fahrers`execute`-Operation startet implizit eine Sitzung und eine Transaktion in dieser Sitzung. Jede Anweisung, die Sie im Lambda-Ausdruck ausführen, wird in die Transaktion eingeschlossen. Nachdem alle Anweisungen ausgeführt wurden, schreibt der Treiber die Transaktion automatisch fest. Wenn eine Anweisung fehlschlägt, nachdem das automatische Wiederholungslimit erschöpft ist, wird die Transaktion abgebrochen.

Verbreiten von Ausnahmeregelungen in einer Transaktion

Wenn Sie mehrere Anweisungen pro Transaktion ausführen, empfehlen wir im Allgemeinen nicht, dass Sie Ausnahmen innerhalb der Transaktion abfangen und übergeben.

Beispielsweise fängt das folgende Programm in Java jede Instance von `RuntimeException` ab, protokolliert den Fehler und fährt fort. Dieses Codebeispiel wird als schlechte Praxis angesehen, da die Transaktion sogar dann erfolgreich ist, wenn die `UPDATE`-Anweisung fehlschlägt. Daher geht der Client möglicherweise davon aus, dass das Update erfolgreich war, auch dies nicht der Fall war.

> **Warning**
>
> Verwenden Sie nicht dieses Codebeispiel. Er wird bereitgestellt, um ein Anti-Pattern-Beispiel zu zeigen, das als schlechte Praxis gilt.

```
// DO NOT USE this code example because it is considered bad practice
public static void main(final String... args) {
    ConnectToLedger.getDriver().execute(txn -> {
        final Result selectTableResult = txn.execute("SELECT * FROM Vehicle WHERE VIN
 ='123456789'");
        // Catching an error inside the transaction is an anti-pattern because the
 operation might
        // not succeed.
        // In this example, the transaction succeeds even when the update statement
 fails.
        // So, the client might assume that the update succeeded when it didn't.
        try {
            processResults(selectTableResult);
            String model = // some code that extracts the model
            final Result updateResult = txn.execute("UPDATE Vehicle SET model = ? WHERE
 VIN = '123456789'",
                    Constants.MAPPER.writeValueAsIonValue(model));
        } catch (RuntimeException e) {
            log.error("Exception when updating the Vehicle table {}", e.getMessage());
        }
    });
    log.info("Vehicle table updated successfully.");
}
```

Propagiere stattdessen die Ausnahme (sprenge nach oben). Wenn ein Teil der Transaktion fehlschlägt, lassen Sie`execute`bricht die Transaktion ab, damit der Kunde die Ausnahme entsprechend behandeln kann.

Verstehen der Wiederholungsrichtlinie mit dem Treiber in Amazon QLDB

Der Amazon QLDB-Treiber verwendet eine Wiederholungsrichtlinie, um vorübergehende Ausnahmen zu behandeln, indem er eine fehlgeschlagene Transaktion transparent wiederholt. Diese Ausnahmen wie`CapacityExceededException`und`RateExceededException`, korrigieren Sie sich in der Regel nach einem bestimmten Zeitraum. Wenn die Transaktion, die mit der Ausnahme fehlgeschlagen ist, nach einer angemessenen Verzögerung erneut versucht wird, wird sie wahrscheinlich erfolgreich sein. Dies trägt dazu bei, die Stabilität der Anwendung zu verbessern, die QLDB verwendet.

Themen

Arten von retryable Fehlern

Der Fahrer versucht eine Transaktion automatisch, wenn und nur dann eine der folgenden Ausnahmen während eines Vorgangs innerhalb dieser Transaktion auftritt:

- CapacityExcededException— Wird zurückgegeben, wenn die Anforderung die Verarbeitungskapazität des Buchs überschreitet.
- InvalidSessionException— Wird zurückgegeben, wenn eine Sitzung nicht mehr gültig ist oder wenn die Sitzung nicht existiert.
- LimitExceededException— Wird zurückgegeben, wenn ein Ressourcenlimit wie die Anzahl aktiver Sitzungen überschritten wird.
- ocConflictException— Wird zurückgegeben, wenn eine Transaktion aufgrund eines Fehlers in der Verifizierungsphase von nicht in das Journal geschrieben werden kannOptimistic Concurrency Control(OCC).
- RateExceedException— Wird zurückgegeben, wenn die Anforderungsrate den erlaubten Durchsatz überschreitet.

Standard-Richtlinie für die Wiederholung von

Die Wiederholungsrichtlinie besteht aus einer Wiederholungsbedingung und einer Backoff-Strategie. Die Wiederholungsbedingung legt fest, wann eine Transaktion erneut versucht werden soll, während die Backoff-Strategie definiert, wie lange gewartet werden soll, bevor die Transaktion erneut versucht wird.

Beim Erstellen einer Instanz des Treibers gibt die Standard-Wiederholungsrichtlinie an, dass Sie es bis zu viermal wiederholen und eine exponentielle Backoffstrategie verwenden. Die exponentielle Backoff-Strategie verwendet eine minimale Verzögerung von 10 Millisekunden und eine maximale Verzögerung von 5000 Millisekunden bei gleichem Jitter. Wenn die Transaktion innerhalb der Wiederholungsrichtlinie nicht erfolgreich übernommen werden kann, empfehlen wir, die Transaktion zu einem anderen Zeitpunkt auszuprobieren.

Das Konzept des exponentiellen Backoffs ist die Verwendung von progressiv längeren Wartezeiten zwischen Wiederholversuchen für aufeinanderfolgende Fehlermeldungen. Weitere Informationen finden Sie im .AWSBlog-PostsExponentielles Backoff und Jitteraus.

Häufige Fehler des Amazon QLDB-Treibers

In diesem Abschnitt werden Laufzeitfehler beschrieben, die vom Amazon QLDB-Treiber bei der Interaktion mit der QLDB-Sitzungs-API ausgelöst werden können.

Im Folgenden finden Sie eine Liste von allgemeinen Ausnahmen, die vom Treiber zurückgegeben werden. Jede Ausnahme enthält die spezifische Fehlermeldung, gefolgt von einer kurzen Beschreibung und Vorschlägen für mögliche Lösungen.

CapacityExceededException

Meldung: Kapazität überschritten

Amazon QLDB lehnte die Anfrage ab, da sie die Verarbeitungskapazität des Ledgers überstieg. QLDB setzt ein internes Skalierungslimit pro Ledger durch, um den Zustand und die Leistung des Dienstes aufrechtzuerhalten. Dieses Limit variiert je nach Workload-Größe der einzelnen Anfrage. Beispielsweise kann eine Anfrage eine erhöhte Arbeitslast haben, wenn sie ineffiziente Datentransaktionen durchführt, wie z. B. Tabellenscans, die aus einer nicht indexqualifizierten Abfrage resultieren.

Wir empfehlen, dass Sie warten, bevor Sie die Anfrage erneut versuchen. Wenn Ihre Anwendung immer wieder auf diese Ausnahme stößt, optimieren Sie Ihre Kontoauszüge und verringern Sie

die Rate und das Volumen der Anfragen, die Sie an das Hauptbuch senden. Beispiele für die Anweisungsoptimierung sind die Ausführung von weniger Anweisungen pro Transaktion und die Optimierung Ihrer Tabellenindizes. Informationen zur Optimierung von Anweisungen und zur Vermeidung von Tabellenscans finden Sie unterOptimieren der Abfrageleistung.

Wir empfehlen außerdem die neueste Version des QLDB-Treiber zu verwenden. Der Treiber hat eine standardmäßige Wiederholungsrichtlinie, die Exponential Backoff und Jitter verwendet, um bei solchen Ausnahmen automatisch Wiederholungsversuche zu starten. Das Konzept des exponentiellen Backoffs besteht darin, für aufeinanderfolgende Fehlerantworten zunehmend längere Wartezeiten zwischen Wiederholungsversuchen zu verwenden.

InvalidSessionException

Meldung: Transaction *`transactionId`* has expired (Transaktion abgelaufen)

Eine Transaktion hat ihre maximale Lebensdauer überschritten. Eine Transaktion kann bis zu 30 Sekunden lang ausgeführt werden, bevor sie bestätigt wird. Nach diesem Timeout-Limit wird jede an der Transaktion geleistete Arbeit abgelehnt und QLDB verwirft die Sitzung. Dieses Limit schützt den Kunden vor Sitzungsverlusten, indem Transaktionen gestartet werden, ohne sie zu bestätigen oder abzubrechen.

Wenn dies eine häufige Ausnahme in Ihrer Anwendung ist, ist es wahrscheinlich, dass die Ausführung von Transaktionen einfach zu lange dauert. Wenn die Transaktionslaufzeit länger als 30 Sekunden dauert, optimieren Sie Ihre Kontoauszüge, um die Transaktionen zu beschleunigen. Beispiele für die Anweisungsoptimierung sind die Ausführung von weniger Anweisungen pro Transaktion und die Optimierung Ihrer Tabellenindizes. Weitere Informationen finden Sie unter Optimieren der Abfrageleistung.

InvalidSessionException

Meldung: Session *`sessionId`* has expired (Sitzung ist abgelaufen)

QLDB hat die Sitzung verworfen, da sie ihre maximale Gesamtlebensdauer überschritten hat. QLDB verwirft Sitzungen nach 13—17 Minuten, unabhängig von einer aktiven Transaktion. Sitzungen können aus einer Reihe von Gründen verloren gehen oder beeinträchtigt werden, wie z. B. Hardwarefehler, Netzwerkfehler oder Anwendungsneustarts. Daher erzwingt QLDB eine maximale Lebensdauer für Sitzungen, um sicherzustellen, dass die Clientsoftware gegen Sitzungsausfälle resistent ist.

Wenn diese Ausnahme auftritt, empfehlen wir, eine neue Sitzung abzurufen und die Transaktion erneut zu versuchen. Wir empfehlen außerdem, die neueste Version des QLDB-Treibers zu verwenden, der den Sitzungspool und seinen Zustand im Namen der Anwendung verwaltet.

InvalidSessionException

Meldung: No such session (Keine solche Sitzung)

Der Client hat versucht, mit QLDB über eine Sitzung zu handeln, die nicht existiert. Unter der Annahme, dass der Client eine Sitzung verwendet, die zuvor existiert hat, ist die Sitzung möglicherweise aus einem der folgenden Gründe nicht mehr vorhanden:

- Wenn eine Sitzung an einem internen Serverausfall beteiligt ist (d. h. ein Fehler mit dem HTTP-Antwortcode 500), kann QLDB sich dafür entscheiden, die Sitzung vollständig zu verwerfen, anstatt dem Kunden die Transaktion mit einer Sitzung mit unsicherem Status zu ermöglichen. Dann schlagen alle Wiederholungsversuche in dieser Sitzung mit diesem Fehler fehl.
- Abgelaufene Sitzungen werden irgendwann von QLDB vergessen. Dann führen alle Versuche, die Sitzung weiter zu verwenden, zu diesem Fehler anstatt des ursprünglichen `InvalidSessionException`.

Wenn diese Ausnahme auftritt, empfehlen wir, eine neue Sitzung abzurufen und die Transaktion erneut zu versuchen. Wir empfehlen außerdem, die neueste Version des QLDB-Treibers zu verwenden, der den Sitzungspool und seinen Zustand im Namen der Anwendung verwaltet.

RateExceededException

Meldung: The rate was exceeded (Die Rate wurde überschritten)

QLDB hat einen Client auf der Grundlage der Identität des Anrufers gedrosselt. QLDB erzwingt die Drosselung pro Region und pro Konto mithilfe eines Token-Bucket-Throttling-Algorithmus. QLDB tut dies, um die Leistung des Dienstes zu unterstützen und um eine faire Nutzung für alle QLDB Kunden sicherzustellen. Beispielsweise kann der Versuch, eine große Anzahl gleichzeitiger Sitzungen mit dem `StartSessionRequest`-Vorgang abzurufen, zu einer Drosselung führen.

Um den korrekten Anwendungszustand aufrechtzuerhalten und eine weitere Drosselung zu verringern, können Sie diese Ausnahme mit exponentiellem Backoff und Jitter wiederholen. Das Konzept des exponentiellen Backoffs besteht darin, für aufeinanderfolgende Fehlerantworten zunehmend längere Wartezeiten zwischen Wiederholungsversuchen zu verwenden. Wir empfehlen die neueste Version des QLDB-Treiber zu verwenden. Der Treiber verfügt über eine standardmäßige Wiederholungsrichtlinie, die exponentielles Backoff und Jitter verwendet, um bei solchen Ausnahmen automatisch einen erneuten Versuch zu starten.

Die neueste Version des QLDB-Treibers kann auch hilfreich sein, wenn Ihre Anwendung ständig von QLDB für`StartSessionRequest` Anrufe gedrosselt wird. Der Treiber verwaltet einen Pool von Sitzungen, die transaktionsübergreifend wiederverwendet werden, was dazu beitragen kann, die Anzahl der`StartSessionRequest` Anrufe zu reduzieren, die Ihre Anwendung tätigt. Um eine Steigerung der API-Drosselungslimits anzufordern, wenden Sie sich an das AWS SupportCenter.

LimitExceededException

Meldung: Exceeded the session limit (Sitzungslimit überschritten)

Ein Ledger hat sein Kontingent (auch als Limit bezeichnet) für die Anzahl aktiver Sitzungen überschritten. Dieses Kontingent ist in Kontingente und Limits in Amazon QLDB definiert. Die Anzahl aktiver Sitzungen eines Ledgers ist schließlich konsistent, und Ledger, die konsistent in der Nähe des Kontingents ausgeführt werden, erfahren diese Ausnahme möglicherweise in regelmäßigen Abständen.

Um die Integrität Ihrer Anwendung aufrechtzuerhalten, empfehlen wir, bei dieser Ausnahme einen erneuten Versuch zu machen. Um diese Ausnahme zu vermeiden, stellen Sie sicher, dass Sie nicht mehr als 1.500 gleichzeitige Sitzungen für ein einzelnes Ledger über alle Clients hinweg konfiguriert haben. Sie können beispielsweise die maxConcurrentTransactionsMethode des Amazon QLDB-Treibers für Java verwenden, um die maximale Anzahl verfügbarer Sitzungen in einer Treiberinstanz zu konfigurieren.

QldbClientException

Meldung: A streamed result is only valid when the parent transaction is open (Ein gestreamtes Ergebnis ist nur gültig, wenn die übergeordnete Transaktion geöffnet ist)

Die Transaktion ist geschlossen und kann nicht verwendet werden, um die Ergebnisse von QLDB abzurufen. Eine Transaktion wird abgeschlossen, wenn sie entweder bestätigt oder storniert wurde.

Diese Ausnahme tritt auf, wenn der Client direkt mit dem`Transaction` Objekt arbeitet und versucht, Ergebnisse von QLDB abzurufen, nachdem er eine Transaktion bestätigt oder abgebrochen hat. Um dieses Problem zu beheben, muss der Kunde die Daten lesen, bevor er die Transaktion abschließt.

Erste Schritte mit Amazon QLDB anhand eines Beispielanwendungs-Tutorials

In diesem Tutorial verwenden Sie den Amazon QLDB-Treiber mit einemAWS SDK, um ein QLDB-Ledger zu erstellen und es mit Beispieldaten zu füllen. Der Treiber ermöglicht es Ihrer Anwendung, mithilfe der Transaktionsdaten-API mit QLDB zu interagieren. DasAWS SDK unterstützt die Interaktion mit der QLDB-Ressourcenmanagement-API.

Das Beispiel-Ledger, das Sie in diesem Szenario anlegen, ist eine Datenbank der Straßenverkehrsbehörde, in der die vollständigen Verlaufsinformationen über Fahrzeugzulassungen nachverfolgt werden. In den folgenden Themen wird erläutert, wie Sie Fahrzeugregistrierungen hinzufügen, ändern und den Änderungsverlauf dieser Zulassungen anzeigen können. Dieser Leitfaden veranschaulicht auch, wie Sie ein Registrierungsdokument kryptografisch verifizieren, und schließt damit ab, wie Ressourcen bereinigt und der Beispiel-Ledger gelöscht wird.

Dieses Beispiel-Programm steht für die folgenden Programmiersprachen zur Verfügung.

Themen

Amazon QLDB Java-Tutorial

In dieser Implementierung der Tutorial-Beispielanwendung verwenden Sie den Amazon QLDB-Treiber mit dem,AWS SDK for Java um ein QLDB-Ledger zu erstellen und es mit Beispieldaten zu füllen.

Während Sie dieses Tutorial durcharbeiten, können Sie sich auf die API-Referenz für Management-API-Operationen auf die AWS SDK for Java-API-Referenz auf die API-Referenz für Management-API-Operationen. Informationen zu Transaktionsdatenoperationen finden Sie in der QLDB-Treiber für Java-API-Referenz.

> **Note**
>
> Gegebenenfalls enthalten einige Tutorialschritte unterschiedliche Befehle oder Codebeispiele für jede unterstützte Hauptversion des QLDB-Treibers für Java.

Themen

Installation der Amazon QLDB Java-Beispielanwendung

In diesem Abschnitt wird beschrieben, wie Sie die bereitgestellte Amazon QLDB-Beispielanwendung für das step-by-step Java-Tutorial installieren und ausführen. Der Anwendungsfall für diese Beispielanwendung ist eine Datenbank der Straßenverkehrsbehörde, mit der die vollständigen Verlaufsinformationen über Fahrzeugzulassungen nachverfolgt werden.

Die DMV-Beispielanwendung für Java ist Open Source im GitHub Repository aws-samples/amazon-qldb-dmv-sample -java.

Voraussetzungen

Vergewissern Sie sich vor der Installation, dass der QLDB-Treiber für Java vollständig istVoraussetzungen. Diese umfasst die folgenden Funktionen:

1. Registrieren beiAWS.
2. Erstellen Sie einen Benutzer mit den entsprechenden QLDB-Berechtigungen. Um alle Schritte in diesem Tutorial abzuschließen, benötigen Sie über die QLDB-API vollen administrativen Zugriff auf Ihre Ledger-Ressource.
3. Wenn Sie eine andere IDE als verwendenAWS Cloud9, installieren Sie Java und gewähren Sie programmatischen Zugriff für die Entwicklung.

Installation

In den folgenden Schritten wird beschrieben, wie Sie die Beispielanwendung mit einer lokalen Entwicklungsumgebung herunterladen und einrichten. Oder Sie können die Einrichtung der Beispielanwendung automatisieren, indem SieAWS Cloud9 als IDE eineAWS CloudFormation Vorlage zur Bereitstellung Ihrer Entwicklungsressourcen verwenden.

Lokale Entwicklungsumgebung

Diese Anweisungen beschreiben, wie Sie die QLDB Java-Beispielanwendung mithilfe Ihrer eigenen Ressourcen und Entwicklungsumgebung herunterladen und installieren.

So laden Sie die Beispielanwendung herunter und führen sie aus

1. Geben Sie den folgenden Befehl ein GitHub.

   2.x

   ```
   git clone https://github.com/aws-samples/amazon-qldb-dmv-sample-java.git
   ```

   1.x

   ```
   git clone -b v1.2.0 https://github.com/aws-samples/amazon-qldb-dmv-sample-
   java.git
   ```

   Dieses Paket enthält die Gradle-Konfiguration und den vollständigen Code aus dem Java-Anleitung.
2. Laden Sie die bereitgestellte Anwendung und führen Sie sie aus.
   - Wenn Sie Eclipse verwenden:
     a. Starten Sie Eclipse und wählen Sie im Menü Eclipse die Optionen File (Datei), Import (Importieren) und dann Existing Gradle Project (Bestehendes Gradle-Projekt).
     b. Durchsuchen Sie das Projekt-Stammverzeichnis und wählen Sie das Anwendungsverzeichnis, das die `build.gradle`-Datei enthält. Wählen Sie dann Finish (Fertig stellen), um die Standard-Gradle-Einstellungen für den Import zu verwenden.

c. Sie können versuchen, das `ListLedgers`-Programm als Beispiel auszuführen. Öffnen Sie das Kontextmenü (rechte Maustaste) für die `ListLedgers.java`-Datei und wählen Sie Run as Java Application (Als Java-Anwendung ausführen) aus.

- Wenn Sie IntelliJ verwenden:

a. Starten Sie IntelliJ und wählen Sie im Menü IntelliJ File (Datei) und dann Open (Öffnen) aus.

b. Durchsuchen Sie das Projekt-Stammverzeichnis und wählen Sie das Anwendungsverzeichnis, das die `build.gradle`-Datei enthält. Klicken Sie anschließend auf OK. Behalten Sie die Standardeinstellungen bei und klicken Sie erneut auf OK.

c. Sie können versuchen, das `ListLedgers`-Programm als Beispiel auszuführen. Öffnen Sie das Kontextmenü (Rechtsklick) für die`ListLedgers.java` Datei und wählen Sie „AusführenListLedgers“.

3. Fahren Sie mit Schritt 1: Erstellen eines neuen Ledger fort, um das Tutorial zu starten und ein Ledger zu erstellen.

AWS Cloud9

In diesen Anweisungen wird beschrieben, wie Sie die Einrichtung der Amazon QLDB-Beispielanwendung für die Fahrzeugregistrierung für Java automatisieren, indem Sie AWS Cloud9sie als IDE verwenden. In diesem Handbuch verwenden Sie eine AWS CloudFormation-Vorlage, um Ihre Entwicklungsressourcen bereitzustellen.

Weitere Informationen zu AWS Cloud9 finden Sie im AWS Cloud9 Benutzerhandbuch. Weitere Informationen zu AWS CloudFormation finden Sie im AWS CloudFormation-Benutzerhandbuch.

Themen

- Teil 1: Stellen Sie Ihre Ressourcen bereit
- Teil 2: Einrichten Ihrer IDE
- Teil 3: Führen Sie die QLDB DMV-Beispielanwendung aus

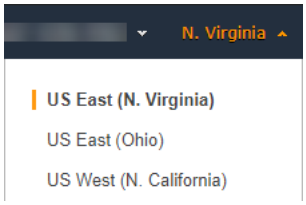
Teil 1: Stellen Sie Ihre Ressourcen bereit

In diesem ersten Schritt verwenden Sie,AWS CloudFormation um die Ressourcen bereitzustellen, die für die Einrichtung Ihrer Entwicklungsumgebung mit der Amazon QLDB-Beispielanwendung erforderlich sind.

Um dieAWS CloudFormation Konsole zu öffnen und die QLDB-Beispielanwendungsvorlage zu laden

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS CloudFormation-Konsole unter https://console.aws.amazon.com/cloudformation.

   Wechseln Sie zu einer Region, die QLDB unterstützt. Eine vollständige Liste finden Sie unter Amazon QLDB-Endpunkte und Kontingente in der Allgemeine AWS-Referenz. Der folgende ScreenshotAWS Management Console zeigt US East (Nord-Virginia)AWS-Region.

2. Wählen Sie auf der AWS CloudFormation-Konsole Create stack (Stack erstellen) und dann With new resources (standard) (Mit neuen Ressourcen) aus.
3. Wählen Sie auf der Seite Create stack (Stack erstellen) unter Specify template (Vorlage angeben) die Option Amazon S3 URL aus.
4. Geben Sie die folgende URL ein und klicken Sie auf Next (Weiter).

   ```
   https://amazon-qldb-assets.s3.amazonaws.com/templates/QLDB-DMV-SampleApp.yml
   ```

5. Geben Sie einen Stack name (Stack-Namen) ein (wie **`qldb-sample-app`**) und wählen Sie Next (Weiter) aus.
6. Sie können Tags nach Bedarf hinzufügen und die Standard-Optionen beibehalten. Wählen Sie anschließend Next (Weiter).
7. Überprüfen Sie Ihre Stack-Einstellungen und klicken Sie auf Create Stack (Stack erstellen). Die Ausführung dieses AWS CloudFormation-Skripts kann einige Minuten dauern.

   Dieses Skript stellt IhrerAWS Cloud9 Umgebung eine zugeordnete Amazon Elastic Compute Cloud (Amazon EC2) -Instance bereit, mit denen Sie die QLDB-Beispielanwendung in diesem Tutorial ausführen. Es klont auch das aws-samples/amazon-qldb-dmv-sample -java-Repository aus GitHub IhrerAWS Cloud9 Entwicklungsumgebung.

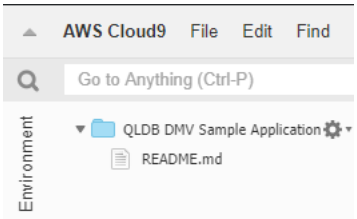
Teil 2: Einrichten Ihrer IDE

In diesem Schritt stellen Sie die Einrichtung Ihrer Cloud-Entwicklungsumgebung fertig. Sie laden ein bereitgestelltes Shell-Skript herunter und führen es aus, um Ihre AWS Cloud9-IDE mit den Abhängigkeiten der Beispielanwendung einzurichten.

So richten Sie Ihre AWS Cloud9-Umgebung ein

1. Öffnen Sie dieAWS Cloud9 Konsole unter https://console.aws.amazon.com/cloud9/.
2. Suchen Sie unter Your environments (Ihre Umgebungen) die Karte für die Umgebung mit der Bezeichnung QLDB DMV Sample Application und wählen Sie Open IDE (IDE öffnen) aus. Es kann einige Minuten dauern, bis Ihre Umgebung lädt, da die zugrunde liegende EC2-Instance gestartet wird.

   Ihre AWS Cloud9-Umgebung ist mit den Systemabhängigkeiten vorkonfiguriert, die Sie benötigen, um das Tutorial auszuführen. Bestätigen Sie im Navigationsbereich Environment (Umgebung) Ihrer Konsole, dass Sie einen Ordner mit der Bezeichnung `QLDB DMV Sample Application` sehen. Der folgende Screenshot derAWS Cloud9 Konsole zeigt den Umgebungsordnerbereich der QLDB DMV Sample Application.

   Wenn Sie keinen Navigationsbereich sehen, blenden Sie die Registerkarte Environment (Umgebung) auf der linken Seite der Konsole ein/aus. Wenn im Bereich keine Ordner angezeigt werden, aktivieren Sie die Option Environment Root anzeigen über das Einstellungssymbol ().
3. Im unteren Bereich Ihrer Konsole sollten Sie ein `bash`-Terminal-Fenster sehen. Wenn Sie dies nicht sehen, wählen Sie New Terminal (Neues Terminal) aus dem Menü Window (Fenster) oben in der Konsole.
4. Laden Sie als Nächstes ein Setup-Skript herunter und führen Sie es aus, um OpenJDK 8 zu installieren, und checken Sie — falls zutreffend — den entsprechenden Branch aus dem Git-Repository aus. Führen Sie in demAWS Cloud9 Terminal, das Sie im vorherigen Schritt erstellt haben:

2.x

```
aws s3 cp s3://amazon-qldb-assets/setup-scripts/dmv-setup-v2.sh .
```

```
sh dmv-setup-v2.sh
```

1.x

```
aws s3 cp s3://amazon-qldb-assets/setup-scripts/dmv-setup.sh .
```

```
sh dmv-setup.sh
```

Nachdem der Vorgang abgeschlossen ist, sollten Sie die folgende Nachricht im Terminal sehen:

```
** DMV Sample App setup completed , enjoy!! **
```

5. Nehmen Sie sich einen Moment Zeit,, den Beispiel-Anwendungscode in AWS Cloud9 zu besuchen, insbesondere im folgenden Verzeichnispfad: `src/main/java/software/amazon/qldb/tutorial`.

Teil 3: Führen Sie die QLDB DMV-Beispielanwendung aus

In diesem Schritt erfahren Sie, wie Sie die Amazon QLDB DMV-Beispielanwendungsaufgaben mit ausführenAWS Cloud9. Zum Ausführen des Beispielcodes gehen Sie zurück zu Ihrem AWS Cloud9-Terminal oder erstellen Sie ein neues Terminal-Fenster, wie Sie es in Teil 2: Einrichten Ihrer IDE gemacht haben.

So führen Sie die Beispielanwendung aus

1. Führen Sie den folgenden Befehl in Ihrem Terminal aus, um zum Projekt-Stammverzeichnis zu wechseln:

```
cd ~/environment/amazon-qldb-dmv-sample-java
```

Stellen Sie sicher, dass Sie die Beispiele im folgenden Verzeichnispfad ausführen.

```
/home/ec2-user/environment/amazon-qldb-dmv-sample-java/
```

2. Der folgende Befehl zeigt die Gradle-Syntax zur Ausführung von einzelnen Aufgaben.

```
./gradlew run -Dtutorial=Task
```

Führen Sie zum Beispiel den folgenden Befehl aus, um alle Ledger in IhrerAWS-Konto und der aktuellen Region aufzulisten.

```
./gradlew run -Dtutorial=ListLedgers
```

3. Fahren Sie mit Schritt 1: Erstellen eines neuen Ledger fort, um das Tutorial zu starten und ein Ledger zu erstellen.
4. (Optional) Wenn Sie das Tutorial abgeschlossen haben, bereinigen Sie Ihre AWS CloudFormation-Ressourcen, wenn Sie sie nicht mehr benötigen.
   a. Öffnen Sie dieAWS CloudFormation Konsole unter https://console.aws.amazon.com/cloudformation und löschen Sie den Stack, den Sie in Teil 1: Bereitstellung Ihrer Ressourcen erstellt haben.
   b. Löschen Sie außerdem den AWS Cloud9-Stack, den die AWS CloudFormation-Vorlage für Sie erstellt hat.

Schritt 1: Erstellen eines neuen Ledger

In diesem Schritt erstellen Sie ein neues Amazon QLDB-Ledger namens`vehicle-registration`.

So erstellen Sie einen neuen Ledger

1. Überprüfen Sie die folgende Datei (`Constants.java`), die konstante Werte enthält, die von allen anderen Programmen in diesem Tutorial verwendet werden.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
```

```
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonSystem;
import com.amazon.ion.system.IonSystemBuilder;
import com.fasterxml.jackson.databind.SerializationFeature;
import com.fasterxml.jackson.dataformat.ion.IonObjectMapper;
import com.fasterxml.jackson.dataformat.ion.ionvalue.IonValueMapper;

/**
 * Constant values used throughout this tutorial.
 */
public final class Constants {
    public static final int RETRY_LIMIT = 4;
    public static final String LEDGER_NAME = "vehicle-registration";
    public static final String STREAM_NAME = "vehicle-registration-stream";
    public static final String VEHICLE_REGISTRATION_TABLE_NAME =
 "VehicleRegistration";
    public static final String VEHICLE_TABLE_NAME = "Vehicle";
    public static final String PERSON_TABLE_NAME = "Person";
    public static final String DRIVERS_LICENSE_TABLE_NAME = "DriversLicense";
    public static final String VIN_INDEX_NAME = "VIN";
    public static final String PERSON_GOV_ID_INDEX_NAME = "GovId";
```

```
    public static final String
 VEHICLE_REGISTRATION_LICENSE_PLATE_NUMBER_INDEX_NAME = "LicensePlateNumber";
    public static final String DRIVER_LICENSE_NUMBER_INDEX_NAME =
 "LicenseNumber";
    public static final String DRIVER_LICENSE_PERSONID_INDEX_NAME = "PersonId";
    public static final String JOURNAL_EXPORT_S3_BUCKET_NAME_PREFIX = "qldb-
tutorial-journal-export";
    public static final String USER_TABLES = "information_schema.user_tables";
    public static final String LEDGER_NAME_WITH_TAGS = "tags";
    public static final IonSystem SYSTEM = IonSystemBuilder.standard().build();
    public static final IonObjectMapper MAPPER = new IonValueMapper(SYSTEM);

    private Constants() { }

    static {
        MAPPER.disable(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS);
    }
}
```

1.x

> **Important**
>
> Für das Amazon Ion-Paket müssen Sie den Namespace `com.amazon.ion` in Ihrer Anwendung verwenden. DasAWS SDK for Java hängt von einem anderen Ion-Paket unter dem Namespace `absoftware.amazon.ion`, aber dies ist ein Legacy-Paket, das nicht mit dem QLDB-Treiber kompatibel ist.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
```

```
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonSystem;
import com.amazon.ion.system.IonSystemBuilder;
import com.fasterxml.jackson.databind.SerializationFeature;
import com.fasterxml.jackson.dataformat.ion.IonObjectMapper;
import com.fasterxml.jackson.dataformat.ion.ionvalue.IonValueMapper;

/**
 * Constant values used throughout this tutorial.
 */
public final class Constants {
    public static final int RETRY_LIMIT = 4;
    public static final String LEDGER_NAME = "vehicle-registration";
    public static final String STREAM_NAME = "vehicle-registration-stream";
    public static final String VEHICLE_REGISTRATION_TABLE_NAME =
 "VehicleRegistration";
    public static final String VEHICLE_TABLE_NAME = "Vehicle";
    public static final String PERSON_TABLE_NAME = "Person";
    public static final String DRIVERS_LICENSE_TABLE_NAME = "DriversLicense";
    public static final String VIN_INDEX_NAME = "VIN";
    public static final String PERSON_GOV_ID_INDEX_NAME = "GovId";
    public static final String
 VEHICLE_REGISTRATION_LICENSE_PLATE_NUMBER_INDEX_NAME = "LicensePlateNumber";
    public static final String DRIVER_LICENSE_NUMBER_INDEX_NAME =
 "LicenseNumber";
    public static final String DRIVER_LICENSE_PERSONID_INDEX_NAME = "PersonId";
    public static final String JOURNAL_EXPORT_S3_BUCKET_NAME_PREFIX = "qldb-
tutorial-journal-export";
    public static final String USER_TABLES = "information_schema.user_tables";
```

```
    public static final String LEDGER_NAME_WITH_TAGS = "tags";
    public static final IonSystem SYSTEM = IonSystemBuilder.standard().build();
    public static final IonObjectMapper MAPPER = new IonValueMapper(SYSTEM);

    private Constants() { }

    static {
        MAPPER.disable(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS);
    }
}
```

> **Note**
>
> Diese`Constants` Klasse beinhaltet eine Instanz der`IonValueMapper` Open-Source-Jackson-Klasse. Sie können diesen Mapper verwenden, um Ihre Amazon Ion-Daten bei Lese- und Schreibtransaktionen zu verarbeiten.

Die `CreateLedger.java`-Datei verfügt auch über eine Abhängigkeit von folgendem Programm (`DescribeLedger.java`), das den aktuellen Status Ihres Ledgers beschreibt.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.model.DescribeLedgerRequest;
import com.amazonaws.services.qldb.model.DescribeLedgerResult;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Describe a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class DescribeLedger {
    public static AmazonQLDB client = CreateLedger.getClient();
    public static final Logger log = LoggerFactory.getLogger(DescribeLedger.class);

    private DescribeLedger() { }

    public static void main(final String... args) {
        try {

            describe(Constants.LEDGER_NAME);

        } catch (Exception e) {
            log.error("Unable to describe a ledger!", e);
        }
    }

    /**
     * Describe a ledger.
     *
     * @param name
     *              Name of the ledger to describe.
     * @return {@link DescribeLedgerResult} from QLDB.
```

```
     */
    public static DescribeLedgerResult describe(final String name) {
        log.info("Let's describe ledger with name: {}...", name);
        DescribeLedgerRequest request = new DescribeLedgerRequest().withName(name);
        DescribeLedgerResult result = client.describeLedger(request);
        log.info("Success. Ledger description: {}", result);
        return result;
    }
}
```

2. Kompilieren Sie das Programm `CreateLedger.java` und führen Sie es aus, um einen Ledger mit dem Namen `vehicle-registration` zu erstellen.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;
```

```
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.AmazonQLDBClientBuilder;
import com.amazonaws.services.qldb.model.CreateLedgerRequest;
import com.amazonaws.services.qldb.model.CreateLedgerResult;
import com.amazonaws.services.qldb.model.DescribeLedgerResult;
import com.amazonaws.services.qldb.model.LedgerState;
import com.amazonaws.services.qldb.model.PermissionsMode;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Create a ledger and wait for it to be active.
 * <p>
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateLedger {
    public static final Logger log =
 LoggerFactory.getLogger(CreateLedger.class);
    public static final Long LEDGER_CREATION_POLL_PERIOD_MS = 10_000L;
    public static String endpoint = null;
    public static String region = null;
    public static AmazonQLDB client = getClient();

    private CreateLedger() {
    }

    /**
     * Build a low-level QLDB client.
     *
     * @return {@link AmazonQLDB} control plane client.
     */
    public static AmazonQLDB getClient() {
        AmazonQLDBClientBuilder builder = AmazonQLDBClientBuilder.standard();
        if (null != endpoint && null != region) {
            builder.setEndpointConfiguration(new
 AwsClientBuilder.EndpointConfiguration(endpoint, region));
        }
        return builder.build();
    }

    public static void main(final String... args) throws Exception {
```

```
        try {
            client = getClient();

            create(Constants.LEDGER_NAME);

            waitForActive(Constants.LEDGER_NAME);

        } catch (Exception e) {
            log.error("Unable to create the ledger!", e);
            throw e;
        }
    }

    /**
     * Create a new ledger with the specified ledger name.
     *
     * @param ledgerName Name of the ledger to be created.
     * @return {@link CreateLedgerResult} from QLDB.
     */
    public static CreateLedgerResult create(final String ledgerName) {
        log.info("Let's create the ledger with name: {}...", ledgerName);
        CreateLedgerRequest request = new CreateLedgerRequest()
                .withName(ledgerName)
                .withPermissionsMode(PermissionsMode.ALLOW_ALL);
        CreateLedgerResult result = client.createLedger(request);
        log.info("Success. Ledger state: {}.", result.getState());
        return result;
    }

    /**
     * Wait for a newly created ledger to become active.
     *
     * @param ledgerName Name of the ledger to wait on.
     * @return {@link DescribeLedgerResult} from QLDB.
     * @throws InterruptedException if thread is being interrupted.
     */
    public static DescribeLedgerResult waitForActive(final String ledgerName)
throws InterruptedException {
        log.info("Waiting for ledger to become active...");
        while (true) {
            DescribeLedgerResult result = DescribeLedger.describe(ledgerName);
            if (result.getState().equals(LedgerState.ACTIVE.name())) {
                log.info("Success. Ledger is active and ready to use.");
                return result;
```

```
            }
            log.info("The ledger is still creating. Please wait...");
            Thread.sleep(LEDGER_CREATION_POLL_PERIOD_MS);
        }
    }
}
```

1.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.AmazonQLDBClientBuilder;
import com.amazonaws.services.qldb.model.CreateLedgerRequest;
import com.amazonaws.services.qldb.model.CreateLedgerResult;
import com.amazonaws.services.qldb.model.DescribeLedgerResult;
import com.amazonaws.services.qldb.model.LedgerState;
```

```
import com.amazonaws.services.qldb.model.PermissionsMode;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Create a ledger and wait for it to be active.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateLedger {
    public static final Logger log =
 LoggerFactory.getLogger(CreateLedger.class);
    public static final Long LEDGER_CREATION_POLL_PERIOD_MS = 10_000L;
    public static AmazonQLDB client = getClient();

    private CreateLedger() { }

    /**
     * Build a low-level QLDB client.
     *
     * @return {@link AmazonQLDB} control plane client.
     */
    public static AmazonQLDB getClient() {
        return AmazonQLDBClientBuilder.standard().build();
    }

    public static void main(final String... args) throws Exception {
        try {

            create(Constants.LEDGER_NAME);

            waitForActive(Constants.LEDGER_NAME);

        } catch (Exception e) {
            log.error("Unable to create the ledger!", e);
            throw e;
        }
    }

    /**
     * Create a new ledger with the specified ledger name.
```

```
     *
     * @param ledgerName
     *              Name of the ledger to be created.
     * @return {@link CreateLedgerResult} from QLDB.
     */
    public static CreateLedgerResult create(final String ledgerName) {
        log.info("Let's create the ledger with name: {}...", ledgerName);
        CreateLedgerRequest request = new CreateLedgerRequest()
                .withName(ledgerName)
                .withPermissionsMode(PermissionsMode.ALLOW_ALL);
        CreateLedgerResult result = client.createLedger(request);
        log.info("Success. Ledger state: {}.", result.getState());
        return result;
    }

    /**
     * Wait for a newly created ledger to become active.
     *
     * @param ledgerName
     *              Name of the ledger to wait on.
     * @return {@link DescribeLedgerResult} from QLDB.
     * @throws InterruptedException if thread is being interrupted.
     */
    public static DescribeLedgerResult waitForActive(final String ledgerName)
 throws InterruptedException {
        log.info("Waiting for ledger to become active...");
        while (true) {
            DescribeLedgerResult result = DescribeLedger.describe(ledgerName);
            if (result.getState().equals(LedgerState.ACTIVE.name())) {
                log.info("Success. Ledger is active and ready to use.");
                return result;
            }
            log.info("The ledger is still creating. Please wait...");
            Thread.sleep(LEDGER_CREATION_POLL_PERIOD_MS);
        }
    }
}
```

> **Note**
>
> - Im `createLedger`-Aufruf müssen Sie einen Ledger-Namen und einen Berechtigungsmodus angeben. Wir empfehlen, den`STANDARD` Berechtigungsmodus, um die Sicherheit Ihrer Ledger-Daten zu maximieren.
> - Wenn Sie einen Ledger erstellen, ist der Löschschutz standardmäßig aktiviert. Dies ist eine Funktion in QLDB, die verhindert, dass Ledger von einem beliebigen Benutzer gelöscht werden. Sie haben die Möglichkeit, den Löschschutz bei der Ledger-Erstellung mithilfe der QLDB-API oder derAWS Command Line Interface (AWS CLI) zu deaktivieren.
> - Optional können Sie auch Tags angeben, die Ihrem Ledger angefügt werden sollen.

Fahren Sie zum Überprüfen der Verbindung mit dem neuen Ledger mit Schritt 2: Testen der Konnektivität des Ledger fort.

Schritt 2: Testen der Konnektivität des Ledger

In diesem Schritt überprüfen Sie, ob Sie über den API-Endpunkt für Transaktionsdaten eine Verbindung zum`vehicle-registration` Ledger in Amazon QLDB herstellen können.

So testen Sie die Verbindung zum Ledger

1. Überprüfen Sie das folgende Programm (`ConnectToLedger.java`), das eine Datensitzungsverbindung zum`vehicle-registration` Ledger herstellt.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
```

```
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.net.URI;
import java.net.URISyntaxException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.services.qldbsession.QldbSessionClient;
import software.amazon.awssdk.services.qldbsession.QldbSessionClientBuilder;
import software.amazon.qldb.QldbDriver;
import software.amazon.qldb.RetryPolicy;

/**
 * Connect to a session for a given ledger using default settings.
 * <p>
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class ConnectToLedger {
    public static final Logger log =
 LoggerFactory.getLogger(ConnectToLedger.class);
    public static AwsCredentialsProvider credentialsProvider;
    public static String endpoint = null;
    public static String ledgerName = Constants.LEDGER_NAME;
    public static String region = null;
    public static QldbDriver driver;
```

```
    private ConnectToLedger() {
    }

    /**
     * Create a pooled driver for creating sessions.
     *
     * @param retryAttempts How many times the transaction will be retried in
     * case of a retryable issue happens like Optimistic Concurrency Control
exception,
     * server side failures or network issues.
     * @return The pooled driver for creating sessions.
     */
    public static QldbDriver createQldbDriver(int retryAttempts) {
        QldbSessionClientBuilder builder = getAmazonQldbSessionClientBuilder();
        return QldbDriver.builder()
                             .ledger(ledgerName)
                             .transactionRetryPolicy(RetryPolicy
                                   .builder()
                                   .maxRetries(retryAttempts)
                                   .build())
                             .sessionClientBuilder(builder)
                             .build();
    }

    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static QldbDriver createQldbDriver() {
        QldbSessionClientBuilder builder = getAmazonQldbSessionClientBuilder();
        return QldbDriver.builder()
            .ledger(ledgerName)
            .transactionRetryPolicy(RetryPolicy.builder()

.maxRetries(Constants.RETRY_LIMIT).build())
            .sessionClientBuilder(builder)
            .build();
    }

    /**
     * Creates a QldbSession builder that is passed to the QldbDriver to connect
to the Ledger.
     *
```

```
     * @return An instance of the AmazonQLDBSessionClientBuilder
     */
    public static QldbSessionClientBuilder getAmazonQldbSessionClientBuilder() {
        QldbSessionClientBuilder builder = QldbSessionClient.builder();
        if (null != endpoint && null != region) {
            try {
                builder.endpointOverride(new URI(endpoint));
            } catch (URISyntaxException e) {
                throw new IllegalArgumentException(e);
            }
        }
        if (null != credentialsProvider) {
            builder.credentialsProvider(credentialsProvider);
        }
        return builder;
    }

    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static QldbDriver getDriver() {
        if (driver == null) {
            driver = createQldbDriver();
        }
        return driver;
    }


    public static void main(final String... args) {
        Iterable<String> tables = ConnectToLedger.getDriver().getTableNames();
        log.info("Existing tables in the ledger:");
        for (String table : tables) {
            log.info("- {} ", table);
        }
    }
}
```

> **Note**
>
> - Um Datenoperationen auf Ihrem Ledger auszuführen, müssen Sie eine Instanz der`QldbDriver` Klasse erstellen, um eine Verbindung zu einem bestimmten Ledger herzustellen. Dies ist ein anderes Client-Objekt als der `AmazonQLDB`-Client, den Sie im vorherigen Schritt zum Erstellen des Ledgers verwendet haben. Dieser vorherige Client wird nur zum Ausführen der Management-API-Operationen verwendet, die in der aufgeführt sindAmazon QLDB API-Referenz.
> - Erstellen Sie zunächst ein `QldbDriver`-Objekt. Beim Erstellen dieses Treibers müssen Sie einen Ledger-Namen angeben.
>
>   Dann können Sie die Methode `execute` dieses Treibers verwenden, um PartiQL-Anweisungen auszuführen.
> - Optional können Sie eine maximale Anzahl von Wiederholungsversuchen für Transaktionsausnahmen angeben. Die`execute` Methode versucht automatisch, Konflikte mit optimistischer Parallelitätskontrolle (OCC) und andere gängige vorübergehende Ausnahmen bis zu diesem konfigurierbaren Limit erneut zu wiederholen. Der Standardwert ist `4`.
>
>   Wenn die Transaktion weiterhin fehlschlägt, nachdem das Limit erreicht ist, löst der Treiber die Ausnahme aus. Weitere Informationen hierzu finden Sie unter Verstehen der Wiederholungsrichtlinie mit dem Treiber in Amazon QLDB.

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
```

```
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.qldbsession.AmazonQLDBSessionClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.PooledQldbDriver;
import software.amazon.qldb.QldbDriver;
import software.amazon.qldb.QldbSession;
import software.amazon.qldb.exceptions.QldbClientException;

/**
 * Connect to a session for a given ledger using default settings.
 * <p>
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class ConnectToLedger {
    public static final Logger log =
 LoggerFactory.getLogger(ConnectToLedger.class);
    public static AWSCredentialsProvider credentialsProvider;
    public static String endpoint = null;
    public static String ledgerName = Constants.LEDGER_NAME;
    public static String region = null;
    private static PooledQldbDriver driver;

    private ConnectToLedger() {
    }
```

```
    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static PooledQldbDriver createQldbDriver() {
        AmazonQLDBSessionClientBuilder builder =
AmazonQLDBSessionClientBuilder.standard();
        if (null != endpoint && null != region) {
            builder.setEndpointConfiguration(new
AwsClientBuilder.EndpointConfiguration(endpoint, region));
        }
        if (null != credentialsProvider) {
            builder.setCredentials(credentialsProvider);
        }
        return PooledQldbDriver.builder()
                .withLedger(ledgerName)
                .withRetryLimit(Constants.RETRY_LIMIT)
                .withSessionClientBuilder(builder)
                .build();
    }

    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static PooledQldbDriver getDriver() {
        if (driver == null) {
            driver = createQldbDriver();
        }
        return driver;
    }

    /**
     * Connect to a ledger through a {@link QldbDriver}.
     *
     * @return {@link QldbSession}.
     */
    public static QldbSession createQldbSession() {
        return getDriver().getSession();
    }
```

```
    public static void main(final String... args) {
        try (QldbSession qldbSession = createQldbSession()) {
            log.info("Listing table names ");
            for (String tableName : qldbSession.getTableNames()) {
                log.info(tableName);
            }
        } catch (QldbClientException e) {
            log.error("Unable to create session.", e);
        }
    }
}
```

Note

- Um Datenoperationen auf Ihrem Ledger auszuführen, müssen Sie eine Instance der `PooledQldbDriver`- oder `QldbDriver`-Klasse erstellen, um eine Verbindung zu einem bestimmten Ledger herzustellen. Dies ist ein anderes Client-Objekt als der `AmazonQLDB`-Client, den Sie im vorherigen Schritt zum Erstellen des Ledgers verwendet haben. Dieser vorherige Client wird nur zum Ausführen der Management-API-Operationen verwendet, die in der aufgeführt sindAmazon QLDB API-Referenz.

  Wir empfehlen die Verwendung,`PooledQldbDriver` es sei denn, Sie müssen einen benutzerdefinierten Sitzungspool mit implementieren`QldbDriver`. Die Standard-Pool-Größe für `PooledQldbDriver` ist die maximale Anzahl offener HTTP-Verbindungen, die der Sitzungs-Client zulässt.

- Erstellen Sie zunächst ein `PooledQldbDriver`-Objekt. Beim Erstellen dieses Treibers müssen Sie einen Ledger-Namen angeben.

  Dann können Sie die Methode `execute` dieses Treibers verwenden, um PartiQL-Anweisungen auszuführen. Sie können auch manuell eine Sitzung aus diesem Pool-Treiberobjekt erstellen und die Methode `execute` der Sitzung verwenden. Eine Sitzung stellt eine einzelne Verbindung mit dem Konto dar.

- Optional können Sie eine maximale Anzahl von Wiederholungsversuchen für Transaktionsausnahmen angeben. Die`execute` Methode versucht automatisch, Konflikte mit optimistischer Parallelitätskontrolle (OCC) und andere gängige vorübergehende Ausnahmen bis zu diesem konfigurierbaren Limit erneut zu wiederholen. Der Standardwert ist `4`.

> Wenn die Transaktion weiterhin fehlschlägt, nachdem das Limit erreicht ist, löst der Treiber die Ausnahme aus. Weitere Informationen hierzu finden Sie unter Verstehen der Wiederholungsrichtlinie mit dem Treiber in Amazon QLDB.

2. Kompilieren Sie das`ConnectToLedger.java` Programm und führen Sie es aus, um die Konnektivität Ihrer Datensitzung zum`vehicle-registration` Ledger zu testen.

Überschreiben desAWS-Region

Die Beispielanwendung stellt in Ihrer Standardeinstellung eine Verbindung zu QLDB herAWS-Region, die Sie wie im Schritt mit den Voraussetzungen beschrieben festlegen könnenFestlegen IhrerAWS Standardanmeldeinformationen und Ihrer Region. Sie können die Region auch ändern, indem Sie die Eigenschaften des QLDB Session Client Builders ändern.

2.x

Im folgenden Codebeispiel wird ein neues `QldbSessionClientBuilder`-Objekt instanziiert.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.qldbsession.QldbSessionClientBuilder;

// This client builder will default to US East (Ohio)
QldbSessionClientBuilder builder = QldbSessionClient.builder()
    .region(Region.US_EAST_2);
```

Sie können die`region` Methode verwenden, um den Code für QLDB in jeder verfügbaren Region auszuführen. Eine vollständige Liste finden Sie unter Amazon QLDB-Endpunkte und Kontingente in der Allgemeine AWS-Referenz.

1.x

Im folgenden Codebeispiel wird ein neues `AmazonQLDBSessionClientBuilder`-Objekt instanziiert.

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.qldbsession.AmazonQLDBSessionClientBuilder;

// This client builder will default to US East (Ohio)
AmazonQLDBSessionClientBuilder builder = AmazonQLDBSessionClientBuilder.standard()
```

```
    .withRegion(Regions.US_EAST_2);
```

Sie können die`withRegion` Methode verwenden, um den Code für QLDB in jeder verfügbaren Region auszuführen. Eine vollständige Liste finden Sie unter Amazon QLDB-Endpunkte und Kontingente in der Allgemeine AWS-Referenz.

Fahren Sie zum Erstellen von Tabellen im `vehicle-registration`-Ledger mit Schritt 3: Tabellen, Indizes und Beispieldaten erstellen fort.

Schritt 3: Tabellen, Indizes und Beispieldaten erstellen

Wenn Ihr Amazon QLDB-Ledger aktiv ist und Verbindungen akzeptiert, können Sie damit beginnen, Tabellen für Daten über Fahrzeuge, deren Besitzer und deren Zulassungsinformationen zu erstellen. Nach dem Erstellen der Tabellen und Indizes können Sie Daten in diese laden.

In diesem Schritt erstellen Sie vier Tabellen im `vehicle-registration`-Ledger:

- `VehicleRegistration`
- `Vehicle`
- `Person`
- `DriversLicense`

Außerdem erzeugen Sie die folgenden Indizes.

Tabellenname	Feld
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Beim Einfügen von Beispieldaten fügen Sie zunächst Dokumente in die `Person`-Tabelle ein. Anschließend verwenden Sie die vom System zugewiesene `id` aus den `Person`-Dokumenten, um die entsprechenden Felder in den relevanten `VehicleRegistration`- und `DriversLicense`-Dokumenten auszufüllen.

> **Tip**
>
> Es hat sich bewährt, den einem Dokument vom System zugewiesenen Schlüssel`id` als Fremdschlüssel zu verwenden. Sie können zwar Felder definieren, die eindeutige Kennungen (z. B. eine Kfz-VIN) sein sollen, die echte eindeutige Kennung eines Dokuments ist aber seine `id`. Dieses Feld ist in den Metadaten des Dokuments enthalten, die Sie in der festgeschriebenen Ansicht (d. h. in der vom System definierten Ansicht der Tabelle) abrufen können.
> Weitere Hinweise zu Ansichten in QLDB finden Sie unterSchlüsselkonzepte. Weitere Informationen zu Metadaten finden Sie unter Metadaten von Dokumenten abfragen.

So richten Sie den Beispiel-Stack ein

1. Überprüfen Sie die folgenden `.java`-Dateien: Diese Modellklassen stellen Dokumente dar, die Sie in den `vehicle-registration`-Tabellen speichern. Sie können in das und vom Amazon Ion-Format serialisiert werden.

   > **Note**
   >
   > Amazon QLDB Dokumentewerden im Ion-Format gespeichert, die eine Obermenge von JSON ist. Sie können also die FasterXML Jackson-Bibliothek verwenden, um die Daten in JSON zu modellieren.

   1. `DriversLicense.java`

      ```
      /*
       * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
       * SPDX-License-Identifier: MIT-0
       *
       * Permission is hereby granted, free of charge, to any person obtaining a copy
      of this
      ```

```
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import software.amazon.qldb.tutorial.model.streams.RevisionData;

import java.time.LocalDate;

/**
 * Represents a driver's license, serializable to (and from) Ion.
 */
public final class DriversLicense implements RevisionData {
    private final String personId;
    private final String licenseNumber;
    private final String licenseType;

    @JsonSerialize(using = IonLocalDateSerializer.class)
    @JsonDeserialize(using = IonLocalDateDeserializer.class)
    private final LocalDate validFromDate;

    @JsonSerialize(using = IonLocalDateSerializer.class)
    @JsonDeserialize(using = IonLocalDateDeserializer.class)
    private final LocalDate validToDate;
```

```
   @JsonCreator
   public DriversLicense(@JsonProperty("PersonId") final String personId,
                         @JsonProperty("LicenseNumber") final String
licenseNumber,
                         @JsonProperty("LicenseType") final String licenseType,
                         @JsonProperty("ValidFromDate") final LocalDate
validFromDate,
                         @JsonProperty("ValidToDate") final LocalDate
validToDate) {
       this.personId = personId;
       this.licenseNumber = licenseNumber;
       this.licenseType = licenseType;
       this.validFromDate = validFromDate;
       this.validToDate = validToDate;
   }

   @JsonProperty("PersonId")
   public String getPersonId() {
       return personId;
   }

   @JsonProperty("LicenseNumber")
   public String getLicenseNumber() {
       return licenseNumber;
   }

   @JsonProperty("LicenseType")
   public String getLicenseType() {
       return licenseType;
   }

   @JsonProperty("ValidFromDate")
   public LocalDate getValidFromDate() {
       return  validFromDate;
   }

   @JsonProperty("ValidToDate")
   public LocalDate getValidToDate() {
       return  validToDate;
   }

   @Override
   public String toString() {
```

```
        return "DriversLicense{" +
                "personId='" + personId + '\'' +
                ", licenseNumber='" + licenseNumber + '\'' +
                ", licenseType='" + licenseType + '\'' +
                ", validFromDate=" + validFromDate +
                ", validToDate=" + validToDate +
                '}';
    }
}
```

2. Person.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import java.time.LocalDate;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
```

```
import com.fasterxml.jackson.databind.annotation.JsonSerialize;

import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.model.streams.RevisionData;

/**
 * Represents a person, serializable to (and from) Ion.
 */
public final class Person implements RevisionData {
    private final String firstName;
    private final String lastName;

    @JsonSerialize(using = IonLocalDateSerializer.class)
    @JsonDeserialize(using = IonLocalDateDeserializer.class)
    private final LocalDate dob;
    private final String govId;
    private final String govIdType;
    private final String address;

    @JsonCreator
    public Person(@JsonProperty("FirstName") final String firstName,
                  @JsonProperty("LastName") final String lastName,
                  @JsonProperty("DOB") final LocalDate dob,
                  @JsonProperty("GovId") final String govId,
                  @JsonProperty("GovIdType") final String govIdType,
                  @JsonProperty("Address") final String address) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.dob = dob;
        this.govId = govId;
        this.govIdType = govIdType;
        this.address = address;
    }

    @JsonProperty("Address")
    public String getAddress() {
        return address;
    }

    @JsonProperty("DOB")
    public LocalDate getDob() {
        return dob;
    }
```

```
    @JsonProperty("FirstName")
    public String getFirstName() {
        return firstName;
    }

    @JsonProperty("LastName")
    public String getLastName() {
        return lastName;
    }

    @JsonProperty("GovId")
    public String getGovId() {
        return govId;
    }

    @JsonProperty("GovIdType")
    public String getGovIdType() {
        return govIdType;
    }

    /**
     * This returns the unique document ID given a specific government ID.
     *
     * @param txn
     *              A transaction executor object.
     * @param govId
     *              The government ID of a driver.
     * @return the unique document ID.
     */
    public static String getDocumentIdByGovId(final TransactionExecutor txn,
final String govId) {
        return SampleData.getDocumentId(txn, Constants.PERSON_TABLE_NAME,
"GovId", govId);
    }

    @Override
    public String toString() {
        return "Person{" +
                "firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", dob=" + dob +
                ", govId='" + govId + '\'' +
                ", govIdType='" + govIdType + '\'' +
```

```
                ", address='" + address + '\'' +
                '}';
    }
}
```

3. VehicleRegistration.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.model.streams.RevisionData;

import java.math.BigDecimal;
import java.time.LocalDate;
```

```
/**
 * Represents a vehicle registration, serializable to (and from) Ion.
 */
public final class VehicleRegistration implements RevisionData {

    private final String vin;
    private final String licensePlateNumber;
    private final String state;
    private final String city;
    private final BigDecimal pendingPenaltyTicketAmount;
    private final LocalDate validFromDate;
    private final LocalDate validToDate;
    private final Owners owners;

    @JsonCreator
    public VehicleRegistration(@JsonProperty("VIN") final String vin,
                               @JsonProperty("LicensePlateNumber") final String
 licensePlateNumber,
                               @JsonProperty("State") final String state,
                               @JsonProperty("City") final String city,
                               @JsonProperty("PendingPenaltyTicketAmount") final
 BigDecimal pendingPenaltyTicketAmount,
                               @JsonProperty("ValidFromDate") final LocalDate
 validFromDate,
                               @JsonProperty("ValidToDate") final LocalDate
 validToDate,
                               @JsonProperty("Owners") final Owners owners) {
        this.vin = vin;
        this.licensePlateNumber = licensePlateNumber;
        this.state = state;
        this.city = city;
        this.pendingPenaltyTicketAmount = pendingPenaltyTicketAmount;
        this.validFromDate = validFromDate;
        this.validToDate = validToDate;
        this.owners = owners;
    }

    @JsonProperty("City")
    public String getCity() {
        return city;
    }

    @JsonProperty("LicensePlateNumber")
```

```
    public String getLicensePlateNumber() {
        return licensePlateNumber;
    }

    @JsonProperty("Owners")
    public Owners getOwners() {
        return owners;
    }

    @JsonProperty("PendingPenaltyTicketAmount")
    public BigDecimal getPendingPenaltyTicketAmount() {
        return pendingPenaltyTicketAmount;
    }

    @JsonProperty("State")
    public String getState() {
        return state;
    }

    @JsonProperty("ValidFromDate")
    @JsonSerialize(using = IonLocalDateSerializer.class)
    @JsonDeserialize(using = IonLocalDateDeserializer.class)
    public LocalDate getValidFromDate() {
        return validFromDate;
    }

    @JsonProperty("ValidToDate")
    @JsonSerialize(using = IonLocalDateSerializer.class)
    @JsonDeserialize(using = IonLocalDateDeserializer.class)
    public LocalDate getValidToDate() {
        return validToDate;
    }

    @JsonProperty("VIN")
    public String getVin() {
        return vin;
    }

    /**
     * Returns the unique document ID of a vehicle given a specific VIN.
     *
     * @param txn
     *              A transaction executor object.
     * @param vin
```

```
     *              The VIN of a vehicle.
     * @return the unique document ID of the specified vehicle.
     */
    public static String getDocumentIdByVin(final TransactionExecutor txn, final
 String vin) {
        return SampleData.getDocumentId(txn,
 Constants.VEHICLE_REGISTRATION_TABLE_NAME, "VIN", vin);
    }

    @Override
    public String toString() {
        return "VehicleRegistration{" +
                "vin='" + vin + '\'' +
                ", licensePlateNumber='" + licensePlateNumber + '\'' +
                ", state='" + state + '\'' +
                ", city='" + city + '\'' +
                ", pendingPenaltyTicketAmount=" + pendingPenaltyTicketAmount +
                ", validFromDate=" + validFromDate +
                ", validToDate=" + validToDate +
                ", owners=" + owners +
                '}';
    }
}
```

4. Vehicle.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
```

```
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import software.amazon.qldb.tutorial.model.streams.RevisionData;

/**
 * Represents a vehicle, serializable to (and from) Ion.
 */
public final class Vehicle implements RevisionData {
    private final String vin;
    private final String type;
    private final int year;
    private final String make;
    private final String model;
    private final String color;

    @JsonCreator
    public Vehicle(@JsonProperty("VIN") final String vin,
                   @JsonProperty("Type") final String type,
                   @JsonProperty("Year") final int year,
                   @JsonProperty("Make") final String make,
                   @JsonProperty("Model") final String model,
                   @JsonProperty("Color") final String color) {
        this.vin = vin;
        this.type = type;
        this.year = year;
        this.make = make;
        this.model = model;
        this.color = color;
    }

    @JsonProperty("Color")
    public String getColor() {
        return color;
    }
```

```
    @JsonProperty("Make")
    public String getMake() {
        return make;
    }

    @JsonProperty("Model")
    public String getModel() {
        return model;
    }

    @JsonProperty("Type")
    public String getType() {
        return type;
    }

    @JsonProperty("VIN")
    public String getVin() {
        return vin;
    }

    @JsonProperty("Year")
    public int getYear() {
        return year;
    }

    @Override
    public String toString() {
        return "Vehicle{" +
                "vin='" + vin + '\'' +
                ", type='" + type + '\'' +
                ", year=" + year +
                ", make='" + make + '\'' +
                ", model='" + model + '\'' +
                ", color='" + color + '\'' +
                '}';
    }
}
```

5. `Owner.java`

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
```

```
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.fasterxml.jackson.annotation.JsonProperty;

/**
 * Represents a vehicle owner, serializable to (and from) Ion.
 */
public final class Owner {
    private final String personId;

    public Owner(@JsonProperty("PersonId") final String personId) {
        this.personId = personId;
    }

    @JsonProperty("PersonId")
    public String getPersonId() {
        return personId;
    }

    @Override
    public String toString() {
        return "Owner{" +
```

```
                "personId='" + personId + '\'' +
                '}';
    }
}
```

6. Owners.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.fasterxml.jackson.annotation.JsonProperty;

import java.util.List;

/**
 * Represents a set of owners for a given vehicle, serializable to (and from)
 Ion.
 */
public final class Owners {
    private final Owner primaryOwner;
```

```
    private final List<Owner> secondaryOwners;

    public Owners(@JsonProperty("PrimaryOwner") final Owner primaryOwner,
                  @JsonProperty("SecondaryOwners") final List<Owner>
 secondaryOwners) {
        this.primaryOwner = primaryOwner;
        this.secondaryOwners = secondaryOwners;
    }

    @JsonProperty("PrimaryOwner")
    public Owner getPrimaryOwner() {
        return primaryOwner;
    }

    @JsonProperty("SecondaryOwners")
    public List<Owner> getSecondaryOwners() {
        return secondaryOwners;
    }

    @Override
    public String toString() {
        return "Owners{" +
                "primaryOwner=" + primaryOwner +
                ", secondaryOwners=" + secondaryOwners +
                '}';
    }
}
```

7. DmlResultDocument.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
```

```
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;

/**
 * Contains information about an individual document inserted or modified
 * as a result of DML.
 */
public class DmlResultDocument {

    private String documentId;

    @JsonCreator
    public DmlResultDocument(@JsonProperty("documentId") final String documentId)
 {
        this.documentId = documentId;
    }

    public String getDocumentId() {
        return documentId;
    }

    @Override
    public String toString() {
        return "DmlResultDocument{"
            + "documentId='" + documentId + '\''
            + '}';
    }
}
```

8. `RevisionData.java`

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model.streams;

/**
 * Allows modeling the content of all revisions as a generic revision data. Used
 * in the {@link Revision} and extended by domain models in {@link
 * software.amazon.qldb.tutorial.model} to make it easier to write the {@link
 * Revision.RevisionDataDeserializer} that must deserialize the {@link
 * Revision#data} from different domain models.
 */
public interface RevisionData { }
```

9. RevisionMetadata.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
```

```
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.amazon.ion.IonInt;
import com.amazon.ion.IonString;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonTimestamp;
import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import com.fasterxml.jackson.dataformat.ion.IonTimestampSerializers;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.Date;
import java.util.Objects;

/**
 * Represents the metadata field of a QLDB Document
 */
public class RevisionMetadata {
    private static final Logger log =
 LoggerFactory.getLogger(RevisionMetadata.class);
    private final String id;
```

```
    private final long version;
    @JsonSerialize(using =
IonTimestampSerializers.IonTimestampJavaDateSerializer.class)
    private final Date txTime;
    private final String txId;

    @JsonCreator
    public RevisionMetadata(@JsonProperty("id") final String id,
                            @JsonProperty("version") final long version,
                            @JsonProperty("txTime") final Date txTime,
                            @JsonProperty("txId") final String txId) {
        this.id = id;
        this.version = version;
        this.txTime = txTime;
        this.txId = txId;
    }

    /**
     * Gets the unique ID of a QLDB document.
     *
     * @return the document ID.
     */
    public String getId() {
        return id;
    }

    /**
     * Gets the version number of the document in the document's modification
history.
     * @return the version number.
     */
    public long getVersion() {
        return version;
    }

    /**
     * Gets the time during which the document was modified.
     *
     * @return the transaction time.
     */
    public Date getTxTime() {
        return txTime;
    }
```

```
    /**
     * Gets the transaction ID associated with this document.
     *
     * @return the transaction ID.
     */
    public String getTxId() {
        return txId;
    }

    public static RevisionMetadata fromIon(final IonStruct ionStruct) {
        if (ionStruct == null) {
            throw new IllegalArgumentException("Metadata cannot be null");
        }
        try {
            IonString id = (IonString) ionStruct.get("id");
            IonInt version = (IonInt) ionStruct.get("version");
            IonTimestamp txTime = (IonTimestamp) ionStruct.get("txTime");
            IonString txId = (IonString) ionStruct.get("txId");
            if (id == null || version == null || txTime == null || txId == null)
{
                throw new IllegalArgumentException("Document is missing required
fields");
            }
            return new RevisionMetadata(id.stringValue(), version.longValue(),
new Date(txTime.getMillis()), txId.stringValue());
        } catch (ClassCastException e) {
            log.error("Failed to parse ion document");
            throw new IllegalArgumentException("Document members are not of the
correct type", e);
        }
    }

    /**
     * Converts a {@link RevisionMetadata} object to a string.
     *
     * @return the string representation of the {@link QldbRevision} object.
     */
    @Override
    public String toString() {
        return "Metadata{"
                + "id='" + id + '\''
                + ", version=" + version
                + ", txTime=" + txTime
                + ", txId='" + txId
```

```
                + '\''
                + '}';
    }

    /**
     * Check whether two {@link RevisionMetadata} objects are equivalent.
     *
     * @return {@code true} if the two objects are equal, {@code false}
 otherwise.
     */
    @Override
    public boolean equals(Object o) {
        if (this == o) { return true; }
        if (o == null || getClass() != o.getClass()) { return false; }
        RevisionMetadata metadata = (RevisionMetadata) o;
        return version == metadata.version
                && id.equals(metadata.id)
                && txTime.equals(metadata.txTime)
                && txId.equals(metadata.txId);
    }

    /**
     * Generate a hash code for the {@link RevisionMetadata} object.
     *
     * @return the hash code.
     */
    @Override
    public int hashCode() {
        // CHECKSTYLE:OFF - Disabling as we are generating a hashCode of multiple
 properties.
        return Objects.hash(id, version, txTime, txId);
        // CHECKSTYLE:ON
    }
}
```

10QldbRevision.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
```

```
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.amazon.ion.IonBlob;
import com.amazon.ion.IonStruct;
import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.Verifier;

import java.io.IOException;
import java.util.Arrays;
import java.util.Objects;

/**
 * Represents a QldbRevision including both user data and metadata.
 */
public final class QldbRevision {
    private static final Logger log =
 LoggerFactory.getLogger(QldbRevision.class);

    private final BlockAddress blockAddress;
    private final RevisionMetadata metadata;
    private final byte[] hash;
```

```
    private final byte[] dataHash;
    private final IonStruct data;

    @JsonCreator
    public QldbRevision(@JsonProperty("blockAddress") final BlockAddress
blockAddress,
                        @JsonProperty("metadata") final RevisionMetadata
metadata,
                        @JsonProperty("hash") final byte[] hash,
                        @JsonProperty("dataHash") final byte[] dataHash,
                        @JsonProperty("data") final IonStruct data) {
        this.blockAddress = blockAddress;
        this.metadata = metadata;
        this.hash = hash;
        this.dataHash = dataHash;
        this.data = data;
    }

    /**
     * Gets the unique ID of a QLDB document.
     *
     * @return the {@link BlockAddress} object.
     */
    public BlockAddress getBlockAddress() {
        return blockAddress;
    }

    /**
     * Gets the metadata of the revision.
     *
     * @return the {@link RevisionMetadata} object.
     */
    public RevisionMetadata getMetadata() {
        return metadata;
    }

    /**
     * Gets the SHA-256 hash value of the revision.
     * This is equivalent to the hash of the revision metadata and data.
     *
     * @return the byte array representing the hash.
     */
    public byte[] getHash() {
        return hash;
```

```
    }

    /**
     * Gets the SHA-256 hash value of the data portion of the revision.
     * This is only present if the revision is redacted.
     *
     * @return the byte array representing the hash.
     */
    public byte[] getDataHash() {
        return dataHash;
    }

    /**
     * Gets the revision data.
     *
     * @return the revision data.
     */
    public IonStruct getData() {
        return data;
    }

    /**
     * Returns true if the revision has been redacted.
     * @return a boolean value representing the redaction status
     * of this revision.
     */
    public Boolean isRedacted() {
        return dataHash != null;
    }

    /**
     * Constructs a new {@link QldbRevision} from an {@link IonStruct}.
     *
     * The specified {@link IonStruct} must include the following fields
     *
     * - blockAddress -- a {@link BlockAddress},
     * - metadata -- a {@link RevisionMetadata},
     * - hash -- the revision's hash calculated by QLDB,
     * - dataHash -- the user data's hash calculated by QLDB (only present if
revision is redacted),
     * - data -- an {@link IonStruct} containing user data in the document.
     *
     * If any of these fields are missing or are malformed, then throws {@link
IllegalArgumentException}.
```

```
    *
    * If the document hash calculated from the members of the specified {@link
IonStruct} does not match
    * the hash member of the {@link IonStruct} then throws {@link
IllegalArgumentException}.
    *
    * @param ionStruct
    *              The {@link IonStruct} that contains a {@link QldbRevision}
object.
    * @return the converted {@link QldbRevision} object.
    * @throws IOException if failed to parse parameter {@link IonStruct}.
    */
   public static QldbRevision fromIon(final IonStruct ionStruct) throws
IOException {
       try {
           BlockAddress blockAddress =
Constants.MAPPER.readValue(ionStruct.get("blockAddress"), BlockAddress.class);
           IonBlob revisionHash = (IonBlob) ionStruct.get("hash");
           IonStruct metadataStruct = (IonStruct) ionStruct.get("metadata");
           IonStruct data = ionStruct.get("data") == null ||
ionStruct.get("data").isNullValue() ?
               null : (IonStruct) ionStruct.get("data");
           IonBlob dataHash = ionStruct.get("dataHash") == null ||
ionStruct.get("dataHash").isNullValue() ?
               null : (IonBlob) ionStruct.get("dataHash");
           if (revisionHash == null || metadataStruct == null) {
               throw new IllegalArgumentException("Document is missing required
fields");
           }
           byte[] dataHashBytes = dataHash != null ? dataHash.getBytes() :
QldbIonUtils.hashIonValue(data);
           verifyRevisionHash(metadataStruct, dataHashBytes,
revisionHash.getBytes());
           RevisionMetadata metadata = RevisionMetadata.fromIon(metadataStruct);
           return new QldbRevision(
                   blockAddress,
                   metadata,
                   revisionHash.getBytes(),
                   dataHash != null ? dataHash.getBytes() : null,
                   data
           );
       } catch (ClassCastException e) {
           log.error("Failed to parse ion document");
```

```
            throw new IllegalArgumentException("Document members are not of the
correct type", e);
        }
    }

    /**
     * Converts a {@link QldbRevision} object to string.
     *
     * @return the string representation of the {@link QldbRevision} object.
     */
    @Override
    public String toString() {
        return "QldbRevision{" +
                "blockAddress=" + blockAddress +
                ", metadata=" + metadata +
                ", hash=" + Arrays.toString(hash) +
                ", dataHash=" + Arrays.toString(dataHash) +
                ", data=" + data +
                '}';
    }

    /**
     * Check whether two {@link QldbRevision} objects are equivalent.
     *
     * @return {@code true} if the two objects are equal, {@code false}
otherwise.
     */
    @Override
    public boolean equals(final Object o) {
        if (this == o) {
            return true;
        }
        if (!(o instanceof QldbRevision)) {
            return false;
        }
        final QldbRevision that = (QldbRevision) o;
        return Objects.equals(getBlockAddress(), that.getBlockAddress())
                && Objects.equals(getMetadata(), that.getMetadata())
                && Arrays.equals(getHash(), that.getHash())
                && Arrays.equals(getDataHash(), that.getDataHash())
                && Objects.equals(getData(), that.getData());
    }

    /**
```

```
     * Create a hash code for the {@link QldbRevision} object.
     *
     * @return the hash code.
     */
    @Override
    public int hashCode() {
        // CHECKSTYLE:OFF - Disabling as we are generating a hashCode of multiple
properties.
        int result = Objects.hash(blockAddress, metadata, data);
        // CHECKSTYLE:ON
        result = 31 * result + Arrays.hashCode(hash);
        return result;
    }

    /**
     * Throws an IllegalArgumentException if the hash of the revision data and
metadata
     * does not match the hash provided by QLDB with the revision.
     */
    public void verifyRevisionHash() {
        // Certain internal-only system revisions only contain a hash which
cannot be
        // further computed. However, these system hashes still participate to
validate
        // the journal block. User revisions will always contain values for all
fields
        // and can therefore have their hash computed.
        if (blockAddress == null && metadata == null && data == null && dataHash
== null) {
            return;
        }

        try {
            IonStruct metadataIon = (IonStruct)
Constants.MAPPER.writeValueAsIonValue(metadata);
            byte[] dataHashBytes = isRedacted() ? dataHash :
QldbIonUtils.hashIonValue(data);
            verifyRevisionHash(metadataIon, dataHashBytes, hash);
        } catch (IOException e) {
            throw new IllegalArgumentException("Could not encode revision
metadata to ion.", e);
        }
    }
```

```
    private static void verifyRevisionHash(IonStruct metadata, byte[] dataHash,
 byte[] expectedHash) {
        byte[] metadataHash = QldbIonUtils.hashIonValue(metadata);
        byte[] candidateHash = Verifier.dot(metadataHash, dataHash);
        if (!Arrays.equals(candidateHash, expectedHash)) {
            throw new IllegalArgumentException("Hash entry of QLDB revision and
 computed hash "
                    + "of QLDB revision do not match");
        }
    }
}
```

11IonLocalDateDeserializer.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.amazon.ion.Timestamp;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.DeserializationContext;
```

```
import com.fasterxml.jackson.databind.JsonDeserializer;

import java.io.IOException;
import java.time.LocalDate;

/**
 * Deserializes [java.time.LocalDate] from Ion.
 */
public class IonLocalDateDeserializer extends JsonDeserializer<LocalDate> {

    @Override
    public LocalDate deserialize(JsonParser jp, DeserializationContext ctxt)
 throws IOException {
        return timestampToLocalDate((Timestamp) jp.getEmbeddedObject());
    }

    private LocalDate timestampToLocalDate(Timestamp timestamp) {
        return LocalDate.of(timestamp.getYear(), timestamp.getMonth(),
 timestamp.getDay());
    }
}
```

12IonLocalDateSerializer.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.amazon.ion.Timestamp;
import com.fasterxml.jackson.core.JsonGenerator;
import com.fasterxml.jackson.databind.SerializerProvider;
import com.fasterxml.jackson.databind.ser.std.StdScalarSerializer;
import com.fasterxml.jackson.dataformat.ion.IonGenerator;

import java.io.IOException;
import java.time.LocalDate;

/**
 * Serializes [java.time.LocalDate] to Ion.
 */
public class IonLocalDateSerializer extends StdScalarSerializer<LocalDate> {

    public IonLocalDateSerializer() {
        super(LocalDate.class);
    }

    @Override
    public void serialize(LocalDate date, JsonGenerator jsonGenerator,
 SerializerProvider serializerProvider) throws IOException {
        Timestamp timestamp = Timestamp.forDay(date.getYear(),
 date.getMonthValue(), date.getDayOfMonth());
        ((IonGenerator) jsonGenerator).writeValue(timestamp);
    }
}
```

2. Überprüfen Sie die folgende Datei (`SampleData.java`), die Daten darstellt, die Sie in die `vehicle-registration`-Tabellen einfügen.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
```

```
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.amazon.ion.IonString;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonValue;
import java.io.IOException;
import java.math.BigDecimal;
import java.text.ParseException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.ConnectToLedger;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.qldb.DmlResultDocument;
import software.amazon.qldb.tutorial.qldb.QldbRevision;
```

```
/**
 * Sample domain objects for use throughout this tutorial.
 */
public final class SampleData {
    public static final DateTimeFormatter DATE_TIME_FORMAT =
 DateTimeFormatter.ofPattern("yyyy-MM-dd");

    public static final List<VehicleRegistration> REGISTRATIONS =
 Collections.unmodifiableList(Arrays.asList(
            new VehicleRegistration("1N4AL11D75C109151", "LEWISR261LL", "WA",
 "Seattle",
                    BigDecimal.valueOf(90.25), convertToLocalDate("2017-08-21"),
 convertToLocalDate("2020-05-11"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("KM8SRDHF6EU074761", "CA762X", "WA", "Kent",
                    BigDecimal.valueOf(130.75),
 convertToLocalDate("2017-09-14"), convertToLocalDate("2020-06-25"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("3HGGK5G53FM761765", "CD820Z", "WA",
 "Everett",
                    BigDecimal.valueOf(442.30),
 convertToLocalDate("2011-03-17"), convertToLocalDate("2021-03-24"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("1HVBBAANXWH544237", "LS477D", "WA",
 "Tacoma",
                    BigDecimal.valueOf(42.20), convertToLocalDate("2011-10-26"),
 convertToLocalDate("2023-09-25"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("1C4RJFAG0FC625797", "TH393F", "WA",
 "Olympia",
                    BigDecimal.valueOf(30.45), convertToLocalDate("2013-09-02"),
 convertToLocalDate("2024-03-19"),
                    new Owners(new Owner(null), Collections.emptyList()))
    ));

    public static final List<Vehicle> VEHICLES =
 Collections.unmodifiableList(Arrays.asList(
            new Vehicle("1N4AL11D75C109151", "Sedan", 2011,  "Audi", "A5",
 "Silver"),
            new Vehicle("KM8SRDHF6EU074761", "Sedan", 2015, "Tesla", "Model S",
 "Blue"),
            new Vehicle("3HGGK5G53FM761765", "Motorcycle", 2011, "Ducati",
 "Monster 1200", "Yellow"),
```

```
            new Vehicle("1HVBBAANXWH544237", "Semi", 2009, "Ford", "F 150",
"Black"),
            new Vehicle("1C4RJFAG0FC625797", "Sedan", 2019, "Mercedes", "CLK
350", "White")
    ));

    public static final List<Person> PEOPLE =
Collections.unmodifiableList(Arrays.asList(
            new Person("Raul", "Lewis", convertToLocalDate("1963-08-19"),
                    "LEWISR261LL", "Driver License",  "1719 University Street,
Seattle, WA, 98109"),
            new Person("Brent", "Logan", convertToLocalDate("1967-07-03"),
                    "LOGANB486CG", "Driver License", "43 Stockert Hollow Road,
Everett, WA, 98203"),
            new Person("Alexis", "Pena", convertToLocalDate("1974-02-10"),
                    "744 849 301", "SSN", "4058 Melrose Street, Spokane Valley,
WA, 99206"),
            new Person("Melvin", "Parker", convertToLocalDate("1976-05-22"),
                    "P626-168-229-765", "Passport", "4362 Ryder Avenue, Seattle,
WA, 98101"),
            new Person("Salvatore", "Spencer", convertToLocalDate("1997-11-15"),
                    "S152-780-97-415-0", "Passport", "4450 Honeysuckle Lane,
Seattle, WA, 98101")
    ));

    public static final List<DriversLicense> LICENSES =
Collections.unmodifiableList(Arrays.asList(
            new DriversLicense(null, "LEWISR261LL", "Learner",
                    convertToLocalDate("2016-12-20"),
convertToLocalDate("2020-11-15")),
            new DriversLicense(null, "LOGANB486CG", "Probationary",
                    convertToLocalDate("2016-04-06"),
convertToLocalDate("2020-11-15")),
            new DriversLicense(null, "744 849 301", "Full",
                    convertToLocalDate("2017-12-06"),
convertToLocalDate("2022-10-15")),
            new DriversLicense(null, "P626-168-229-765", "Learner",
                    convertToLocalDate("2017-08-16"),
convertToLocalDate("2021-11-15")),
            new DriversLicense(null, "S152-780-97-415-0", "Probationary",
                    convertToLocalDate("2015-08-15"),
convertToLocalDate("2021-08-21"))
    ));
```

```
    private SampleData() { }

    /**
     * Converts a date string with the format 'yyyy-MM-dd' into a {@link
java.util.Date} object.
     *
     * @param date
     *              The date string to convert.
     * @return {@link java.time.LocalDate} or null if there is a {@link
ParseException}
     */
    public static synchronized LocalDate convertToLocalDate(String date) {
        return LocalDate.parse(date, DATE_TIME_FORMAT);
    }

    /**
     * Convert the result set into a list of IonValues.
     *
     * @param result
     *              The result set to convert.
     * @return a list of IonValues.
     */
    public static List<IonValue> toIonValues(Result result) {
        final List<IonValue> valueList = new ArrayList<>();
        result.iterator().forEachRemaining(valueList::add);
        return valueList;
    }

    /**
     * Get the document ID of a particular document.
     *
     * @param txn
     *              A transaction executor object.
     * @param tableName
     *              Name of the table containing the document.
     * @param identifier
     *              The identifier used to narrow down the search.
     * @param value
     *              Value of the identifier.
     * @return the list of document IDs in the result set.
     */
    public static String getDocumentId(final TransactionExecutor txn, final
String tableName,
```

```
                                        final String identifier, final String
value) {
       try {
           final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(value));
           final String query = String.format("SELECT metadata.id FROM
_ql_committed_%s AS p WHERE p.data.%s = ?",
                   tableName, identifier);
           Result result = txn.execute(query, parameters);
           if (result.isEmpty()) {
               throw new IllegalStateException("Unable to retrieve document ID
using " + value);
           }
           return getStringValueOfStructField((IonStruct)
result.iterator().next(), "id");
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }

   /**
    * Get the document by ID.
    *
    * @param tableName
    *              Name of the table to insert documents into.
    * @param documentId
    *              The unique ID of a document in the Person table.
    * @return a {@link QldbRevision} object.
    * @throws IllegalStateException if failed to convert parameter into {@link
IonValue}.
    */
   public static QldbRevision getDocumentById(String tableName, String
documentId) {
       try {
           final IonValue ionValue =
Constants.MAPPER.writeValueAsIonValue(documentId);
           Result result = ConnectToLedger.getDriver().execute(txn -> {
               return txn.execute("SELECT c.* FROM _ql_committed_" + tableName
+ " AS c BY docId "
                                   + "WHERE docId = ?", ionValue);
           });
           if (result.isEmpty()) {
               throw new IllegalStateException("Unable to retrieve document by
id " + documentId + " in table " + tableName);
```

```
            }
            return Constants.MAPPER.readValue(result.iterator().next(),
QldbRevision.class);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Return a list of modified document IDs as strings from a DML {@link
Result}.
     *
     * @param result
     *              The result set from a DML operation.
     * @return the list of document IDs modified by the operation.
     */
    public static List<String> getDocumentIdsFromDmlResult(final Result result)
{
        final List<String> strings = new ArrayList<>();
        result.iterator().forEachRemaining(row ->
strings.add(getDocumentIdFromDmlResultDocument(row)));
        return strings;
    }

    /**
     * Convert the given DML result row's document ID to string.
     *
     * @param dmlResultDocument
     *              The {@link IonValue} representing the results of a DML
operation.
     * @return a string of document ID.
     */
    public static String getDocumentIdFromDmlResultDocument(final IonValue
dmlResultDocument) {
        try {
            DmlResultDocument result =
Constants.MAPPER.readValue(dmlResultDocument, DmlResultDocument.class);
            return result.getDocumentId();
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
```

```
    * Get the String value of a given {@link IonStruct} field name.
    * @param struct the {@link IonStruct} from which to get the value.
    * @param fieldName the name of the field from which to get the value.
    * @return the String value of the field within the given {@link IonStruct}.
    */
   public static String getStringValueOfStructField(final IonStruct struct,
final String fieldName) {
       return ((IonString) struct.get(fieldName)).stringValue();
   }

   /**
    * Return a copy of the given driver's license with updated person Id.
    *
    * @param oldLicense
    *              The old driver's license to update.
    * @param personId
    *              The PersonId of the driver.
    * @return the updated {@link DriversLicense}.
    */
   public static DriversLicense updatePersonIdDriversLicense(final
DriversLicense oldLicense, final String personId) {
       return new DriversLicense(personId, oldLicense.getLicenseNumber(),
oldLicense.getLicenseType(),
               oldLicense.getValidFromDate(), oldLicense.getValidToDate());
   }

   /**
    * Return a copy of the given vehicle registration with updated person Id.
    *
    * @param oldRegistration
    *              The old vehicle registration to update.
    * @param personId
    *              The PersonId of the driver.
    * @return the updated {@link VehicleRegistration}.
    */
   public static VehicleRegistration updateOwnerVehicleRegistration(final
VehicleRegistration oldRegistration,
                                                                    final
String personId) {
       return new VehicleRegistration(oldRegistration.getVin(),
oldRegistration.getLicensePlateNumber(),
               oldRegistration.getState(), oldRegistration.getCity(),
oldRegistration.getPendingPenaltyTicketAmount(),
```

```
                oldRegistration.getValidFromDate(),
 oldRegistration.getValidToDate(),
                new Owners(new Owner(personId), Collections.emptyList()));
    }
}
```

1.x

> **Important**
>
> Für das Amazon Ion-Paket müssen Sie den Namespace `com.amazon.ion` in Ihrer Anwendung verwenden. DasAWS SDK for Java hängt von einem anderen Ion-Paket unter dem Namespace `absoftware.amazon.ion`, aber dies ist ein Legacy-Paket, das nicht mit dem QLDB-Treiber kompatibel ist.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
```

```
package software.amazon.qldb.tutorial.model;

import com.amazon.ion.IonString;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonValue;
import software.amazon.qldb.QldbSession;
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.qldb.DmlResultDocument;
import software.amazon.qldb.tutorial.qldb.QldbRevision;

import java.io.IOException;

import java.math.BigDecimal;
import java.text.ParseException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

/**
 * Sample domain objects for use throughout this tutorial.
 */
public final class SampleData {
    public static final DateTimeFormatter DATE_TIME_FORMAT =
 DateTimeFormatter.ofPattern("yyyy-MM-dd");

    public static final List<VehicleRegistration> REGISTRATIONS =
 Collections.unmodifiableList(Arrays.asList(
            new VehicleRegistration("1N4AL11D75C109151", "LEWISR261LL", "WA",
 "Seattle",
                    BigDecimal.valueOf(90.25), convertToLocalDate("2017-08-21"),
 convertToLocalDate("2020-05-11"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("KM8SRDHF6EU074761", "CA762X", "WA", "Kent",
                    BigDecimal.valueOf(130.75),
 convertToLocalDate("2017-09-14"), convertToLocalDate("2020-06-25"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("3HGGK5G53FM761765", "CD820Z", "WA",
 "Everett",
```

```
                    BigDecimal.valueOf(442.30),
convertToLocalDate("2011-03-17"), convertToLocalDate("2021-03-24"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("1HVBBAANXWH544237", "LS477D", "WA",
"Tacoma",
                    BigDecimal.valueOf(42.20), convertToLocalDate("2011-10-26"),
convertToLocalDate("2023-09-25"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("1C4RJFAG0FC625797", "TH393F", "WA",
"Olympia",
                    BigDecimal.valueOf(30.45), convertToLocalDate("2013-09-02"),
convertToLocalDate("2024-03-19"),
                    new Owners(new Owner(null), Collections.emptyList()))
    ));

    public static final List<Vehicle> VEHICLES =
Collections.unmodifiableList(Arrays.asList(
            new Vehicle("1N4AL11D75C109151", "Sedan", 2011,  "Audi", "A5",
"Silver"),
            new Vehicle("KM8SRDHF6EU074761", "Sedan", 2015, "Tesla", "Model S",
"Blue"),
            new Vehicle("3HGGK5G53FM761765", "Motorcycle", 2011, "Ducati",
"Monster 1200", "Yellow"),
            new Vehicle("1HVBBAANXWH544237", "Semi", 2009, "Ford", "F 150",
"Black"),
            new Vehicle("1C4RJFAG0FC625797", "Sedan", 2019, "Mercedes", "CLK
350", "White")
    ));

    public static final List<Person> PEOPLE =
Collections.unmodifiableList(Arrays.asList(
            new Person("Raul", "Lewis", convertToLocalDate("1963-08-19"),
                    "LEWISR261LL", "Driver License",  "1719 University Street,
Seattle, WA, 98109"),
            new Person("Brent", "Logan", convertToLocalDate("1967-07-03"),
                    "LOGANB486CG", "Driver License", "43 Stockert Hollow Road,
Everett, WA, 98203"),
            new Person("Alexis", "Pena", convertToLocalDate("1974-02-10"),
                    "744 849 301", "SSN", "4058 Melrose Street, Spokane Valley,
WA, 99206"),
            new Person("Melvin", "Parker", convertToLocalDate("1976-05-22"),
                    "P626-168-229-765", "Passport", "4362 Ryder Avenue, Seattle,
WA, 98101"),
            new Person("Salvatore", "Spencer", convertToLocalDate("1997-11-15"),
```

```
                    "S152-780-97-415-0", "Passport", "4450 Honeysuckle Lane,
Seattle, WA, 98101")
    ));

    public static final List<DriversLicense> LICENSES =
Collections.unmodifiableList(Arrays.asList(
            new DriversLicense(null, "LEWISR261LL", "Learner",
                    convertToLocalDate("2016-12-20"),
convertToLocalDate("2020-11-15")),
            new DriversLicense(null, "LOGANB486CG", "Probationary",
                    convertToLocalDate("2016-04-06"),
convertToLocalDate("2020-11-15")),
            new DriversLicense(null, "744 849 301", "Full",
                    convertToLocalDate("2017-12-06"),
convertToLocalDate("2022-10-15")),
            new DriversLicense(null, "P626-168-229-765", "Learner",
                    convertToLocalDate("2017-08-16"),
convertToLocalDate("2021-11-15")),
            new DriversLicense(null, "S152-780-97-415-0", "Probationary",
                    convertToLocalDate("2015-08-15"),
convertToLocalDate("2021-08-21"))
    ));

    private SampleData() { }

    /**
     * Converts a date string with the format 'yyyy-MM-dd' into a {@link
java.util.Date} object.
     *
     * @param date
     *              The date string to convert.
     * @return {@link LocalDate} or null if there is a {@link ParseException}
     */
    public static synchronized LocalDate convertToLocalDate(String date) {
        return LocalDate.parse(date, DATE_TIME_FORMAT);
    }

    /**
     * Convert the result set into a list of IonValues.
     *
     * @param result
     *              The result set to convert.
     * @return a list of IonValues.
     */
```

```
    public static List<IonValue> toIonValues(Result result) {
        final List<IonValue> valueList = new ArrayList<>();
        result.iterator().forEachRemaining(valueList::add);
        return valueList;
    }

    /**
     * Get the document ID of a particular document.
     *
     * @param txn
     *              A transaction executor object.
     * @param tableName
     *              Name of the table containing the document.
     * @param identifier
     *              The identifier used to narrow down the search.
     * @param value
     *              Value of the identifier.
     * @return the list of document IDs in the result set.
     */
    public static String getDocumentId(final TransactionExecutor txn, final
String tableName,
                                       final String identifier, final String
value) {
        try {
            final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(value));
            final String query = String.format("SELECT metadata.id FROM
_ql_committed_%s AS p WHERE p.data.%s = ?",
                    tableName, identifier);
            Result result = txn.execute(query, parameters);
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to retrieve document ID
using " + value);
            }
            return getStringValueOfStructField((IonStruct)
result.iterator().next(), "id");
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Get the document by ID.
     *
```

```
     * @param qldbSession
     *              A QLDB session.
     * @param tableName
     *              Name of the table to insert documents into.
     * @param documentId
     *              The unique ID of a document in the Person table.
     * @return a {@link QldbRevision} object.
     * @throws IllegalStateException if failed to convert parameter into {@link
IonValue}.
     */
    public static QldbRevision getDocumentById(QldbSession qldbSession, String
tableName, String documentId) {
        try {
            final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(documentId));
            final String query = String.format("SELECT c.* FROM _ql_committed_%s
AS c BY docId WHERE docId = ?", tableName);
            Result result = qldbSession.execute(query, parameters);
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to retrieve document by
id " + documentId + " in table " + tableName);
            }
            return Constants.MAPPER.readValue(result.iterator().next(),
QldbRevision.class);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Return a list of modified document IDs as strings from a DML {@link
Result}.
     *
     * @param result
     *              The result set from a DML operation.
     * @return the list of document IDs modified by the operation.
     */
    public static List<String> getDocumentIdsFromDmlResult(final Result result)
{
        final List<String> strings = new ArrayList<>();
        result.iterator().forEachRemaining(row ->
strings.add(getDocumentIdFromDmlResultDocument(row)));
        return strings;
    }
```

```
   /**
    * Convert the given DML result row's document ID to string.
    *
    * @param dmlResultDocument
    *              The {@link IonValue} representing the results of a DML
operation.
    * @return a string of document ID.
    */
   public static String getDocumentIdFromDmlResultDocument(final IonValue
dmlResultDocument) {
       try {
           DmlResultDocument result =
Constants.MAPPER.readValue(dmlResultDocument, DmlResultDocument.class);
           return result.getDocumentId();
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }

   /**
    * Get the String value of a given {@link IonStruct} field name.
    * @param struct the {@link IonStruct} from which to get the value.
    * @param fieldName the name of the field from which to get the value.
    * @return the String value of the field within the given {@link IonStruct}.
    */
   public static String getStringValueOfStructField(final IonStruct struct,
final String fieldName) {
       return ((IonString) struct.get(fieldName)).stringValue();
   }

   /**
    * Return a copy of the given driver's license with updated person Id.
    *
    * @param oldLicense
    *              The old driver's license to update.
    * @param personId
    *              The PersonId of the driver.
    * @return the updated {@link DriversLicense}.
    */
   public static DriversLicense updatePersonIdDriversLicense(final
DriversLicense oldLicense, final String personId) {
       return new DriversLicense(personId, oldLicense.getLicenseNumber(),
oldLicense.getLicenseType(),
```

```
                oldLicense.getValidFromDate(), oldLicense.getValidToDate());
    }

    /**
     * Return a copy of the given vehicle registration with updated person Id.
     *
     * @param oldRegistration
     *              The old vehicle registration to update.
     * @param personId
     *              The PersonId of the driver.
     * @return the updated {@link VehicleRegistration}.
     */
    public static VehicleRegistration updateOwnerVehicleRegistration(final
 VehicleRegistration oldRegistration,
                                                                     final
 String personId) {
        return new VehicleRegistration(oldRegistration.getVin(),
 oldRegistration.getLicensePlateNumber(),
                oldRegistration.getState(), oldRegistration.getCity(),
 oldRegistration.getPendingPenaltyTicketAmount(),
                oldRegistration.getValidFromDate(),
 oldRegistration.getValidToDate(),
                new Owners(new Owner(personId), Collections.emptyList()));
    }
}
```

> **Note**
>
> - Diese Klasse verwendet Ion-Bibliotheken, um Hilfsmethoden bereitzustellen, die Ihre Daten in das und aus dem Ion-Format konvertieren.
> - Die `getDocumentId`-Methode führt eine Abfrage für eine Tabelle mit dem Präfix `_ql_committed_` aus. Dies ist ein reserviertes Präfix, das angibt, dass Sie die bestätigte Ansicht einer Tabelle abfragen möchten. In dieser Ansicht sind Ihre Daten im `data`-Feld verschachtelt und Metadaten sind im `metadata`-Feld verschachtelt.

3. Kompilieren Sie das folgende Programm (`CreateTable.java`) und führen Sie es aus, um die zuvor genannten Tabellen zu erstellen.

2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Create tables in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
```

```
 */
public final class CreateTable {
    public static final Logger log = LoggerFactory.getLogger(CreateTable.class);

    private CreateTable() { }

    /**
     * Registrations, vehicles, owners, and licenses tables being created in a
 single transaction.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to be created.
     * @return the number of tables created.
     */
    public static int createTable(final TransactionExecutor txn, final String
 tableName) {
        log.info("Creating the '{}' table...", tableName);
        final String createTable = String.format("CREATE TABLE %s", tableName);
        final Result result = txn.execute(createTable);
        log.info("{} table created successfully.", tableName);
        return SampleData.toIonValues(result).size();
    }

    public static void main(final String... args) {
        ConnectToLedger.getDriver().execute(txn -> {
            createTable(txn, Constants.DRIVERS_LICENSE_TABLE_NAME);
            createTable(txn, Constants.PERSON_TABLE_NAME);
            createTable(txn, Constants.VEHICLE_TABLE_NAME);
            createTable(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME);
        });
    }
}
```

1.x

```
/*
 * Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
```

```
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Create tables in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateTable {
    public static final Logger log = LoggerFactory.getLogger(CreateTable.class);

    private CreateTable() { }

    /**
     * Registrations, vehicles, owners, and licenses tables being created in a
 single transaction.
```

```
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to be created.
     * @return the number of tables created.
     */
    public static int createTable(final TransactionExecutor txn, final String
 tableName) {
        log.info("Creating the '{}' table...", tableName);
        final String createTable = String.format("CREATE TABLE %s", tableName);
        final Result result = txn.execute(createTable);
        log.info("{} table created successfully.", tableName);
        return SampleData.toIonValues(result).size();
    }

    public static void main(final String... args) {
        ConnectToLedger.getDriver().execute(txn -> {
            createTable(txn, Constants.DRIVERS_LICENSE_TABLE_NAME);
            createTable(txn, Constants.PERSON_TABLE_NAME);
            createTable(txn, Constants.VEHICLE_TABLE_NAME);
            createTable(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME);
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
    }
}
```

> **Note**
>
> Dieses Programm verdeutlicht, wie ein `TransactionExecutor`-Lambda an die `execute`-Methode übergeben wird. In diesem Beispiel führen Sie mehrere`CREATE TABLE` PartiQL-Anweisungen in einer einzigen Transaktion aus, indem Sie einen Lambda-Ausdruck verwenden.
>
> Die`execute` Methode startet implizit eine Transaktion, führt alle Anweisungen im Lambda aus und überträgt die Transaktion dann automatisch.

4. Kompilieren Sie das folgende Programm (`CreateIndex.java`) und führen Sie es aus, um wie zuvor beschrieben Indizes für die Tabellen zu erstellen.

2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Create indexes on tables in a particular ledger.
 *
 * This code expects that you have AWS credentials setup per:
```

```
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateIndex {
    public static final Logger log = LoggerFactory.getLogger(CreateIndex.class);

    private CreateIndex() { }

    /**
     * In this example, create indexes for registrations and vehicles tables.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to be created.
     * @param indexAttribute
     *              The index attribute to use.
     * @return the number of tables created.
     */
    public static int createIndex(final TransactionExecutor txn, final String
 tableName, final String indexAttribute) {
        log.info("Creating an index on {}...", indexAttribute);
        final String createIndex = String.format("CREATE INDEX ON %s (%s)",
 tableName, indexAttribute);
        final Result r = txn.execute(createIndex);
        return SampleData.toIonValues(r).size();
    }

    public static void main(final String... args) {
        ConnectToLedger.getDriver().execute(txn -> {
            createIndex(txn, Constants.PERSON_TABLE_NAME,
 Constants.PERSON_GOV_ID_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_TABLE_NAME,
 Constants.VIN_INDEX_NAME);
            createIndex(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.DRIVER_LICENSE_NUMBER_INDEX_NAME);
            createIndex(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.DRIVER_LICENSE_PERSONID_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 Constants.VIN_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,

 Constants.VEHICLE_REGISTRATION_LICENSE_PLATE_NUMBER_INDEX_NAME);
        });
```

```
        log.info("Indexes created successfully!");
    }
}
```

1.x

```
/*
 * Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Create indexes on tables in a particular ledger.
```

```
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateIndex {
    public static final Logger log = LoggerFactory.getLogger(CreateIndex.class);

    private CreateIndex() { }

    /**
     * In this example, create indexes for registrations and vehicles tables.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to be created.
     * @param indexAttribute
     *              The index attribute to use.
     * @return the number of tables created.
     */
    public static int createIndex(final TransactionExecutor txn, final String
 tableName, final String indexAttribute) {
        log.info("Creating an index on {}...", indexAttribute);
        final String createIndex = String.format("CREATE INDEX ON %s (%s)",
 tableName, indexAttribute);
        final Result r = txn.execute(createIndex);
        return SampleData.toIonValues(r).size();
    }

    public static void main(final String... args) {
        ConnectToLedger.getDriver().execute(txn -> {
            createIndex(txn, Constants.PERSON_TABLE_NAME,
 Constants.PERSON_GOV_ID_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_TABLE_NAME,
 Constants.VIN_INDEX_NAME);
            createIndex(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.DRIVER_LICENSE_NUMBER_INDEX_NAME);
            createIndex(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.DRIVER_LICENSE_PERSONID_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 Constants.VIN_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
```

```
 Constants.VEHICLE_REGISTRATION_LICENSE_PLATE_NUMBER_INDEX_NAME);
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
        log.info("Indexes created successfully!");
    }
}
```

5. Kompilieren Sie das folgende Programm (`InsertDocument.java`) und führen Sie es aus, um die Beispieldaten in Ihre Tabellen einzufügen.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
```

```
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.DriversLicense;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.model.VehicleRegistration;

/**
 * Insert documents into a table in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class InsertDocument {
    public static final Logger log =
 LoggerFactory.getLogger(InsertDocument.class);

    private InsertDocument() { }

    /**
     * Insert the given list of documents into the specified table and return
 the document IDs of the inserted documents.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to insert documents into.
     * @param documents
     *              List of documents to insert into the specified table.
     * @return a list of document IDs.
     * @throws IllegalStateException if failed to convert documents into an
 {@link IonValue}.
     */
    public static List<String> insertDocuments(final TransactionExecutor txn,
 final String tableName,
                                               final List documents) {
        log.info("Inserting some documents in the {} table...", tableName);
        try {
```

```
            final String query = String.format("INSERT INTO %s ?", tableName);
            final IonValue ionDocuments =
Constants.MAPPER.writeValueAsIonValue(documents);

            return SampleData.getDocumentIdsFromDmlResult(txn.execute(query,
ionDocuments));
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Update PersonIds in driver's licenses and in vehicle registrations using
document IDs.
     *
     * @param documentIds
     *              List of document IDs representing the PersonIds in
DriversLicense and PrimaryOwners in VehicleRegistration.
     * @param licenses
     *              List of driver's licenses to update.
     * @param registrations
     *              List of registrations to update.
     */
    public static void updatePersonId(final List<String> documentIds, final
List<DriversLicense> licenses,
                                      final List<VehicleRegistration>
registrations) {
        for (int i = 0; i < documentIds.size(); ++i) {
            DriversLicense license = SampleData.LICENSES.get(i);
            VehicleRegistration registration = SampleData.REGISTRATIONS.get(i);
            licenses.add(SampleData.updatePersonIdDriversLicense(license,
documentIds.get(i)));

registrations.add(SampleData.updateOwnerVehicleRegistration(registration,
documentIds.get(i)));
        }
    }

    public static void main(final String... args) {
        final List<DriversLicense> newDriversLicenses = new ArrayList<>();
        final List<VehicleRegistration> newVehicleRegistrations = new
ArrayList<>();
        ConnectToLedger.getDriver().execute(txn -> {
```

```
            List<String> documentIds = insertDocuments(txn,
 Constants.PERSON_TABLE_NAME, SampleData.PEOPLE);
            updatePersonId(documentIds, newDriversLicenses,
 newVehicleRegistrations);
            insertDocuments(txn, Constants.VEHICLE_TABLE_NAME,
 SampleData.VEHICLES);
            insertDocuments(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
                    Collections.unmodifiableList(newVehicleRegistrations));
            insertDocuments(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
                    Collections.unmodifiableList(newDriversLicenses));
        });
        log.info("Documents inserted successfully!");
    }
}
```

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
```

```
package software.amazon.qldb.tutorial;

import com.amazon.ion.IonValue;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.QldbSession;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.DriversLicense;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.model.VehicleRegistration;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Insert documents into a table in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class InsertDocument {
    public static final Logger log =
 LoggerFactory.getLogger(InsertDocument.class);

    private InsertDocument() { }

    /**
     * Insert the given list of documents into the specified table and return
 the document IDs of the inserted documents.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to insert documents into.
     * @param documents
     *              List of documents to insert into the specified table.
     * @return a list of document IDs.
     * @throws IllegalStateException if failed to convert documents into an
 {@link IonValue}.
     */
```

```
   public static List<String> insertDocuments(final TransactionExecutor txn,
final String tableName,
                                              final List documents) {
       log.info("Inserting some documents in the {} table...", tableName);
       try {
           final String statement = String.format("INSERT INTO %s ?",
tableName);
           final IonValue ionDocuments =
Constants.MAPPER.writeValueAsIonValue(documents);
           final List<IonValue> parameters =
Collections.singletonList(ionDocuments);
           return SampleData.getDocumentIdsFromDmlResult(txn.execute(statement,
parameters));
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }

   /**
    * Update PersonIds in driver's licenses and in vehicle registrations using
document IDs.
    *
    * @param documentIds
    *              List of document IDs representing the PersonIds in
DriversLicense and PrimaryOwners in VehicleRegistration.
    * @param licenses
    *              List of driver's licenses to update.
    * @param registrations
    *              List of registrations to update.
    */
   public static void updatePersonId(final List<String> documentIds, final
List<DriversLicense> licenses,
                                     final List<VehicleRegistration>
registrations) {
       for (int i = 0; i < documentIds.size(); ++i) {
           DriversLicense license = SampleData.LICENSES.get(i);
           VehicleRegistration registration = SampleData.REGISTRATIONS.get(i);
           licenses.add(SampleData.updatePersonIdDriversLicense(license,
documentIds.get(i)));

registrations.add(SampleData.updateOwnerVehicleRegistration(registration,
documentIds.get(i)));
       }
   }
```

```
    public static void main(final String... args) {
        final List<DriversLicense> newDriversLicenses = new ArrayList<>();
        final List<VehicleRegistration> newVehicleRegistrations = new
 ArrayList<>();
        ConnectToLedger.getDriver().execute(txn -> {
            List<String> documentIds = insertDocuments(txn,
 Constants.PERSON_TABLE_NAME, SampleData.PEOPLE);
            updatePersonId(documentIds, newDriversLicenses,
 newVehicleRegistrations);
            insertDocuments(txn, Constants.VEHICLE_TABLE_NAME,
 SampleData.VEHICLES);
            insertDocuments(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
                    Collections.unmodifiableList(newVehicleRegistrations));
            insertDocuments(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
                    Collections.unmodifiableList(newDriversLicenses));
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
        log.info("Documents inserted successfully!");
    }
}
```

> **Note**
>
> - Dieses Programm veranschaulicht, wie die `execute`-Methode mit parametrisierten Werten aufgerufen wird. Sie können außerdem eine Liste mit Datenparametern des Typs `IonValue` an die auszuführende PartiQL-Anweisung übergeben. Verwenden Sie ein Fragezeichen (?) als Variablenplatzhalter in Ihrer Anweisungszeichenfolge.
> - Wenn eine `INSERT`-Anweisung erfolgreich ist, wird die `id` jedes eingefügten Dokuments zurückgegeben.

Als Nächstes können Sie mit `SELECT`-Anweisungen Daten aus den Tabellen im `vehicle-registration`-Ledger lesen. Fahren Sie mit Schritt 4: Abfragen der Tabellen in einem Ledger fort.

Schritt 4: Abfragen der Tabellen in einem Ledger

Nachdem Sie Tabellen in einem Amazon QLDB-Ledger erstellt und sie mit Daten geladen haben, können Sie Abfragen ausführen, um die Fahrzeugregistrierungsdaten zu überprüfen, die Sie

gerade eingegeben haben. QLDB verwendet PartiQL als Abfragesprache und Amazon Ion als dokumentorientiertes Datenmodell.

PartiQL ist eine quelloffene, SQL-kompatible Abfragesprache, die für die Arbeit mit Ion erweitert wurde. Mit PartiSQL können Sie Ihre Daten mit vertrauten SQL-Operatoren einfügen, abfragen und verwalten. Amazon Ion ist eine Obermenge von JSON. Ion ist ein dokumentenbasiertes Open-Source-Datenformat, das Ihnen die Flexibilität bietet, strukturierte, halbstrukturierte und verschachtelte Daten zu speichern und zu verarbeiten.

In diesem Schritt verwenden Sie `SELECT`-Anweisungen zum Lesen von Daten aus den Tabellen im `vehicle-registration`-Ledger.

> **Warning**
>
> Wenn Sie eine Abfrage in QLDB ohne indizierte Suche ausführen, wird ein vollständiger Tabellenscan aufgerufen. PartiQL unterstützt solche Abfragen, da es SQL-kompatibel ist. Führen Sie jedoch keine Tabellenscans für produktive Anwendungsfälle in QLDB aus. Tabellenscans können bei großen Tabellen zu Leistungsproblemen führen, einschließlich Parallelitätskonflikten und Transaktions-Timeouts.
> Um Tabellenscans zu vermeiden, müssen Sie Anweisungen mit einer`WHERE` Prädikatklausel ausführen, die einen Gleichheitsoperator für ein indiziertes Feld oder eine Dokument-ID verwenden, z. B.`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`. Weitere Informationen finden Sie unter Optimieren der Abfrageleistung.

So fragen Sie die Tabellen ab

- Kompilieren Sie das folgende Programm (`FindVehicles.java`) und führen Sie es aus, um alle Fahrzeuge abzufragen, die unter einer Person in Ihrem Ledger registriert sind.

  2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
```

```
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.io.IOException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Find all vehicles registered under a person.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class FindVehicles {
    public static final Logger log =
 LoggerFactory.getLogger(FindVehicles.class);

    private FindVehicles() { }
```

```
    /**
     * Find vehicles registered under a driver using their government ID.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param govId
     *              The government ID of the owner.
     * @throws IllegalStateException if failed to convert parameters into {@link
 IonValue}.
     */
    public static void findVehiclesForOwner(final TransactionExecutor txn, final
 String govId) {
        try {
            final String documentId = Person.getDocumentIdByGovId(txn, govId);
            final String query = "SELECT v FROM Vehicle AS v INNER JOIN
 VehicleRegistration AS r "
                    + "ON v.VIN = r.VIN WHERE r.Owners.PrimaryOwner.PersonId
 = ?";

            final Result result = txn.execute(query,
 Constants.MAPPER.writeValueAsIonValue(documentId));
            log.info("List of Vehicles for owner with GovId: {}...", govId);
            ScanTable.printDocuments(result);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final Person person = SampleData.PEOPLE.get(0);
        ConnectToLedger.getDriver().execute(txn -> {
            findVehiclesForOwner(txn, person.getGovId());
        });
    }
}
```

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
```

```
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.io.IOException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Find all vehicles registered under a person.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class FindVehicles {
```

```
    public static final Logger log =
 LoggerFactory.getLogger(FindVehicles.class);

    private FindVehicles() { }

    /**
     * Find vehicles registered under a driver using their government ID.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param govId
     *              The government ID of the owner.
     * @throws IllegalStateException if failed to convert parameters into {@link
 IonValue}.
     */
    public static void findVehiclesForOwner(final TransactionExecutor txn, final
 String govId) {
        try {
            final String documentId = Person.getDocumentIdByGovId(txn, govId);
            final String query = "SELECT v FROM Vehicle AS v INNER JOIN
 VehicleRegistration AS r "
                    + "ON v.VIN = r.VIN WHERE r.Owners.PrimaryOwner.PersonId
 = ?";

            final Result result = txn.execute(query,
 Constants.MAPPER.writeValueAsIonValue(documentId));
            log.info("List of Vehicles for owner with GovId: {}...", govId);
            ScanTable.printDocuments(result);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final Person person = SampleData.PEOPLE.get(0);
        ConnectToLedger.getDriver().execute(txn -> {
            findVehiclesForOwner(txn, person.getGovId());
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
    }
}
```

> **Note**
>
> Zunächst fragt dieses Programm die `Person`-Tabelle nach dem Dokument mit `GovId` `LEWISR261LL` ab, um sein `id`-Metadatenfeld abzurufen.
> Anschließend verwendet es dieses Dokument `id` als Fremdschlüssel, um die `VehicleRegistration`-Tabelle nach `PrimaryOwner.PersonId` abzufragen. Es tritt führt außerdem `VehicleRegistration` mit der `Vehicle`-Tabelle im `VIN`-Feld zusammen.

Weitere Informationen zum Ändern von Dokumenten in den Tabellen im `vehicle-registration`-Ledger finden Sie unter Schritt 5: Dokumente in einem Ledger ändern.

Schritt 5: Dokumente in einem Ledger ändern

Da Sie nun über Daten verfügen, mit denen Sie arbeiten können, können Sie damit beginnen, Änderungen an Dokumenten im`vehicle-registration` Hauptbuch in Amazon QLDB vorzunehmen. In diesem Schritt verdeutlichen die folgenden Codebeispiele, wie Sie Data Manipulation Language (DML)-Anweisungen ausführen. Diese Anweisungen aktualisieren den primären Eigentümer eines Fahrzeugs und fügen einem anderen Fahrzeug einen sekundären Eigentümer hinzu.

So ändern Sie Dokumente

1. Kompilieren Sie das folgende Programm (`TransferVehicleOwnership.java`) und führen Sie es aus, um den primären Eigentümer eines Fahrzeugs in Ihrem Ledger mit VIN `1N4AL11D75C109151` zu aktualisieren.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
```

```
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonReader;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonReaderBuilder;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.LinkedHashMap;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Owner;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Find primary owner for a particular vehicle's VIN.
 * Transfer to another primary owner for a particular vehicle's VIN.
 *
```

```
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class TransferVehicleOwnership {
    public static final Logger log =
 LoggerFactory.getLogger(TransferVehicleOwnership.class);

    private TransferVehicleOwnership() { }

    /**
     * Query a driver's information using the given ID.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param documentId
     *              The unique ID of a document in the Person table.
     * @return a {@link Person} object.
     * @throws IllegalStateException if failed to convert parameter into {@link
 IonValue}.
     */
    public static Person findPersonFromDocumentId(final TransactionExecutor txn,
 final String documentId) {
        try {
            log.info("Finding person for documentId: {}...", documentId);
            final String query = "SELECT p.* FROM Person AS p BY pid WHERE pid
 = ?";

            Result result = txn.execute(query,
 Constants.MAPPER.writeValueAsIonValue(documentId));
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to find person with ID:
 " + documentId);
            }

            return Constants.MAPPER.readValue(result.iterator().next(),
 Person.class);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Find the primary owner for the given VIN.
```

```
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @return a {@link Person} object.
     * @throws IllegalStateException if failed to convert parameter into {@link
IonValue}.
     */
    public static Person findPrimaryOwnerForVehicle(final TransactionExecutor
txn, final String vin) {
        try {
            log.info("Finding primary owner for vehicle with Vin: {}...", vin);
            final String query = "SELECT Owners.PrimaryOwner.PersonId FROM
VehicleRegistration AS v WHERE v.VIN = ?";
            final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
            Result result = txn.execute(query, parameters);
            final List<IonStruct> documents = ScanTable.toIonStructs(result);
            ScanTable.printDocuments(documents);
            if (documents.isEmpty()) {
                throw new IllegalStateException("Unable to find registrations
with VIN: " + vin);
            }

            final IonReader reader =
IonReaderBuilder.standard().build(documents.get(0));
            final String personId = Constants.MAPPER.readValue(reader,
LinkedHashMap.class).get("PersonId").toString();
            return findPersonFromDocumentId(txn, personId);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Update the primary owner for a vehicle registration with the given
documentId.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param documentId
```

```
     *              New PersonId for the primary owner.
     * @throws IllegalStateException if no vehicle registration was found using
the given document ID and VIN, or if failed
     * to convert parameters into {@link IonValue}.
     */
    public static void updateVehicleRegistration(final TransactionExecutor txn,
final String vin, final String documentId) {
        try {
            log.info("Updating primary owner for vehicle with Vin: {}...", vin);
            final String query = "UPDATE VehicleRegistration AS v SET
v.Owners.PrimaryOwner = ? WHERE v.VIN = ?";

            final List<IonValue> parameters = new ArrayList<>();
            parameters.add(Constants.MAPPER.writeValueAsIonValue(new
Owner(documentId)));
            parameters.add(Constants.MAPPER.writeValueAsIonValue(vin));

            Result result = txn.execute(query, parameters);
            ScanTable.printDocuments(result);
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to transfer vehicle,
could not find registration.");
            } else {
                log.info("Successfully transferred vehicle with VIN '{}' to new
owner.", vin);
            }
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String vin = SampleData.VEHICLES.get(0).getVin();
        final String primaryOwnerGovId = SampleData.PEOPLE.get(0).getGovId();
        final String newPrimaryOwnerGovId = SampleData.PEOPLE.get(1).getGovId();

        ConnectToLedger.getDriver().execute(txn -> {
            final Person primaryOwner = findPrimaryOwnerForVehicle(txn, vin);
            if (!primaryOwner.getGovId().equals(primaryOwnerGovId)) {
                // Verify the primary owner.
                throw new IllegalStateException("Incorrect primary owner
identified for vehicle, unable to transfer.");
            }
```

```
            final String newOwner = Person.getDocumentIdByGovId(txn,
 newPrimaryOwnerGovId);
            updateVehicleRegistration(txn, vin, newOwner);
        });
        log.info("Successfully transferred vehicle ownership!");
    }
}
```

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonReader;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonReaderBuilder;
```

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.LinkedHashMap;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Owner;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Find primary owner for a particular vehicle's VIN.
 * Transfer to another primary owner for a particular vehicle's VIN.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class TransferVehicleOwnership {
    public static final Logger log =
 LoggerFactory.getLogger(TransferVehicleOwnership.class);

    private TransferVehicleOwnership() { }

    /**
     * Query a driver's information using the given ID.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param documentId
     *              The unique ID of a document in the Person table.
     * @return a {@link Person} object.
     * @throws IllegalStateException if failed to convert parameter into {@link
 IonValue}.
     */
    public static Person findPersonFromDocumentId(final TransactionExecutor txn,
 final String documentId) {
        try {
            log.info("Finding person for documentId: {}...", documentId);
```

```
            final String query = "SELECT p.* FROM Person AS p BY pid WHERE pid
= ?";

            Result result = txn.execute(query,
Constants.MAPPER.writeValueAsIonValue(documentId));
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to find person with ID:
" + documentId);
            }

            return Constants.MAPPER.readValue(result.iterator().next(),
Person.class);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Find the primary owner for the given VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @return a {@link Person} object.
     * @throws IllegalStateException if failed to convert parameter into {@link
IonValue}.
     */
    public static Person findPrimaryOwnerForVehicle(final TransactionExecutor
txn, final String vin) {
        try {
            log.info("Finding primary owner for vehicle with Vin: {}...", vin);
            final String query = "SELECT Owners.PrimaryOwner.PersonId FROM
VehicleRegistration AS v WHERE v.VIN = ?";
            final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
            Result result = txn.execute(query, parameters);
            final List<IonStruct> documents = ScanTable.toIonStructs(result);
            ScanTable.printDocuments(documents);
            if (documents.isEmpty()) {
                throw new IllegalStateException("Unable to find registrations
with VIN: " + vin);
            }
```

```
            final IonReader reader =
IonReaderBuilder.standard().build(documents.get(0));
            final String personId = Constants.MAPPER.readValue(reader,
LinkedHashMap.class).get("PersonId").toString();
            return findPersonFromDocumentId(txn, personId);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Update the primary owner for a vehicle registration with the given
documentId.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param documentId
     *              New PersonId for the primary owner.
     * @throws IllegalStateException if no vehicle registration was found using
the given document ID and VIN, or if failed
     * to convert parameters into {@link IonValue}.
     */
    public static void updateVehicleRegistration(final TransactionExecutor txn,
final String vin, final String documentId) {
        try {
            log.info("Updating primary owner for vehicle with Vin: {}...", vin);
            final String query = "UPDATE VehicleRegistration AS v SET
v.Owners.PrimaryOwner = ? WHERE v.VIN = ?";

            final List<IonValue> parameters = new ArrayList<>();
            parameters.add(Constants.MAPPER.writeValueAsIonValue(new
Owner(documentId)));
            parameters.add(Constants.MAPPER.writeValueAsIonValue(vin));

            Result result = txn.execute(query, parameters);
            ScanTable.printDocuments(result);
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to transfer vehicle,
could not find registration.");
            } else {
                log.info("Successfully transferred vehicle with VIN '{}' to new
owner.", vin);
```

```
            }
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String vin = SampleData.VEHICLES.get(0).getVin();
        final String primaryOwnerGovId = SampleData.PEOPLE.get(0).getGovId();
        final String newPrimaryOwnerGovId = SampleData.PEOPLE.get(1).getGovId();

        ConnectToLedger.getDriver().execute(txn -> {
            final Person primaryOwner = findPrimaryOwnerForVehicle(txn, vin);
            if (!primaryOwner.getGovId().equals(primaryOwnerGovId)) {
                // Verify the primary owner.
                throw new IllegalStateException("Incorrect primary owner
 identified for vehicle, unable to transfer.");
            }

            final String newOwner = Person.getDocumentIdByGovId(txn,
 newPrimaryOwnerGovId);
            updateVehicleRegistration(txn, vin, newOwner);
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
        log.info("Successfully transferred vehicle ownership!");
    }
}
```

2. Kompilieren Sie das folgende Programm (`AddSecondaryOwner.java`) und führen Sie es aus, um dem Fahrzeug mit VIN `KM8SRDHF6EU074761` in Ihrem Ledger einen sekundären Eigentümer hinzuzufügen.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
```

```
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.io.IOException;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Owner;
import software.amazon.qldb.tutorial.model.Owners;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Finds and adds secondary owners for a vehicle.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class AddSecondaryOwner {
```

```
   public static final Logger log =
LoggerFactory.getLogger(AddSecondaryOwner.class);

   private AddSecondaryOwner() { }

   /**
    * Check whether a secondary owner has already been registered for the given
VIN.
    *
    * @param txn
    *              The {@link TransactionExecutor} for lambda execute.
    * @param vin
    *              Unique VIN for a vehicle.
    * @param secondaryOwnerId
    *              The secondary owner to add.
    * @return {@code true} if the given secondary owner has already been
registered, {@code false} otherwise.
    * @throws IllegalStateException if failed to convert VIN to an {@link
IonValue}.
    */
   public static boolean isSecondaryOwnerForVehicle(final TransactionExecutor
txn, final String vin,
                                                    final String
secondaryOwnerId) {
       try {
           log.info("Finding secondary owners for vehicle with VIN: {}...",
vin);
           final String query = "SELECT Owners.SecondaryOwners FROM
VehicleRegistration AS v WHERE v.VIN = ?";
           final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
           final Result result = txn.execute(query, parameters);
           final Iterator<IonValue> itr = result.iterator();
           if (!itr.hasNext()) {
               return false;
           }

           final Owners owners = Constants.MAPPER.readValue(itr.next(),
Owners.class);
           if (null != owners.getSecondaryOwners()) {
               for (Owner owner : owners.getSecondaryOwners()) {
                   if (secondaryOwnerId.equalsIgnoreCase(owner.getPersonId()))
{
                       return true;
```

```
                    }
                }
            }

            return false;
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Adds a secondary owner for the specified VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param secondaryOwner
     *              The secondary owner to add.
     * @throws IllegalStateException if failed to convert parameter into an
{@link IonValue}.
     */
    public static void addSecondaryOwnerForVin(final TransactionExecutor txn,
final String vin,
                                               final String secondaryOwner) {
        try {
            log.info("Inserting secondary owner for vehicle with VIN: {}...",
vin);
            final String query = String.format("FROM VehicleRegistration AS v
WHERE v.VIN = ?" +
                    "INSERT INTO v.Owners.SecondaryOwners VALUE ?");
            final IonValue newOwner = Constants.MAPPER.writeValueAsIonValue(new
Owner(secondaryOwner));
            final IonValue vinAsIonValue =
Constants.MAPPER.writeValueAsIonValue(vin);
            Result result = txn.execute(query, vinAsIonValue, newOwner);
            log.info("VehicleRegistration Document IDs which had secondary
owners added: ");
            ScanTable.printDocuments(result);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }
```

```
    public static void main(final String... args) {
        final String vin = SampleData.VEHICLES.get(1).getVin();
        final String govId = SampleData.PEOPLE.get(0).getGovId();

        ConnectToLedger.getDriver().execute(txn -> {
            final String documentId = Person.getDocumentIdByGovId(txn, govId);
            if (isSecondaryOwnerForVehicle(txn, vin, documentId)) {
                log.info("Person with ID {} has already been added as a
 secondary owner of this vehicle.", govId);
            } else {
                addSecondaryOwnerForVin(txn, vin, documentId);
            }
        });
        log.info("Secondary owners successfully updated.");
    }
}
```

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

```
 */

package software.amazon.qldb.tutorial;

import java.io.IOException;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Owner;
import software.amazon.qldb.tutorial.model.Owners;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Finds and adds secondary owners for a vehicle.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class AddSecondaryOwner {
    public static final Logger log =
 LoggerFactory.getLogger(AddSecondaryOwner.class);

    private AddSecondaryOwner() { }

    /**
     * Check whether a secondary owner has already been registered for the given
 VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param secondaryOwnerId
     *              The secondary owner to add.
```

```
    * @return {@code true} if the given secondary owner has already been
registered, {@code false} otherwise.
    * @throws IllegalStateException if failed to convert VIN to an {@link
IonValue}.
    */
   public static boolean isSecondaryOwnerForVehicle(final TransactionExecutor
txn, final String vin,
                                                    final String
secondaryOwnerId) {
       try {
           log.info("Finding secondary owners for vehicle with VIN: {}...",
vin);
           final String query = "SELECT Owners.SecondaryOwners FROM
VehicleRegistration AS v WHERE v.VIN = ?";
           final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
           final Result result = txn.execute(query, parameters);
           final Iterator<IonValue> itr = result.iterator();
           if (!itr.hasNext()) {
               return false;
           }

           final Owners owners = Constants.MAPPER.readValue(itr.next(),
Owners.class);
           if (null != owners.getSecondaryOwners()) {
               for (Owner owner : owners.getSecondaryOwners()) {
                   if (secondaryOwnerId.equalsIgnoreCase(owner.getPersonId()))
{
                       return true;
                   }
               }
           }

           return false;
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }

   /**
    * Adds a secondary owner for the specified VIN.
    *
    * @param txn
    *              The {@link TransactionExecutor} for lambda execute.
```

```
     * @param vin
     *              Unique VIN for a vehicle.
     * @param secondaryOwner
     *              The secondary owner to add.
     * @throws IllegalStateException if failed to convert parameter into an
 {@link IonValue}.
     */
    public static void addSecondaryOwnerForVin(final TransactionExecutor txn,
 final String vin,
                                               final String secondaryOwner) {
        try {
            log.info("Inserting secondary owner for vehicle with VIN: {}...",
 vin);
            final String query = String.format("FROM VehicleRegistration AS v
 WHERE v.VIN = '%s' " +
                    "INSERT INTO v.Owners.SecondaryOwners VALUE ?", vin);
            final IonValue newOwner = Constants.MAPPER.writeValueAsIonValue(new
 Owner(secondaryOwner));
            Result result = txn.execute(query, newOwner);
            log.info("VehicleRegistration Document IDs which had secondary
 owners added: ");
            ScanTable.printDocuments(result);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String vin = SampleData.VEHICLES.get(1).getVin();
        final String govId = SampleData.PEOPLE.get(0).getGovId();

        ConnectToLedger.getDriver().execute(txn -> {
            final String documentId = Person.getDocumentIdByGovId(txn, govId);
            if (isSecondaryOwnerForVehicle(txn, vin, documentId)) {
                log.info("Person with ID {} has already been added as a
 secondary owner of this vehicle.", govId);
            } else {
                addSecondaryOwnerForVin(txn, vin, documentId);
            }
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
        log.info("Secondary owners successfully updated.");
    }
}
```

Informationen zum Überprüfen dieser Änderungen im `vehicle-registration`-Ledger finden Sie unter Schritt 6: Den Revisionsverlauf für ein Dokument anzeigen.

Schritt 6: Den Revisionsverlauf für ein Dokument anzeigen

Nach dem Bearbeiten der Zulassungsdaten für ein Fahrzeug im vorherigen Schritt können Sie den Verlauf aller registrierten Eigentümer und alle anderen aktualisierten Felder abfragen. In diesem Schritt führen Sie eine Abfrage des Revisionsverlaufs eines Dokuments in der `VehicleRegistration`-Tabelle in Ihrem `vehicle-registration`-Ledger durch.

So zeigen Sie den Revisionsverlauf an

1. Überprüfen Sie das folgende Programm (`QueryHistory.java`).

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;
```

```
import java.io.IOException;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.model.VehicleRegistration;

/**
 * Query a table's history for a particular set of documents.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class QueryHistory {
    public static final Logger log =
 LoggerFactory.getLogger(QueryHistory.class);
    private static final int THREE_MONTHS = 90;

    private QueryHistory() { }

    /**
     * In this example, query the 'VehicleRegistration' history table to find
 all previous primary owners for a VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              VIN to find previous primary owners for.
     * @param query
     *              The query to find previous primary owners.
     * @throws IllegalStateException if failed to convert document ID to an
 {@link IonValue}.
     */
    public static void previousPrimaryOwners(final TransactionExecutor txn,
 final String vin, final String query) {
```

```
        try {
            final String docId = VehicleRegistration.getDocumentIdByVin(txn,
 vin);

            log.info("Querying the 'VehicleRegistration' table's history using
 VIN: {}...", vin);
            final Result result = txn.execute(query,
 Constants.MAPPER.writeValueAsIonValue(docId));
            ScanTable.printDocuments(result);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String threeMonthsAgo = Instant.now().minus(THREE_MONTHS,
 ChronoUnit.DAYS).toString();
        final String query = String.format("SELECT data.Owners.PrimaryOwner,
 metadata.version "
                                           + "FROM history(VehicleRegistration,
 `%s`) "
                                           + "AS h WHERE h.metadata.id = ?",
 threeMonthsAgo);
        ConnectToLedger.getDriver().execute(txn -> {
            final String vin = SampleData.VEHICLES.get(0).getVin();
            previousPrimaryOwners(txn, vin, query);
        });
        log.info("Successfully queried history.");
    }
}
```

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
```

```
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonValue;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.QldbSession;
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.model.VehicleRegistration;

import java.io.IOException;
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.Collections;
import java.util.List;

/**
 * Query a table's history for a particular set of documents.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class QueryHistory {
    public static final Logger log =
 LoggerFactory.getLogger(QueryHistory.class);
    private static final int THREE_MONTHS = 90;
```

```
   private QueryHistory() { }

   /**
    * In this example, query the 'VehicleRegistration' history table to find
all previous primary owners for a VIN.
    *
    * @param txn
    *              The {@link TransactionExecutor} for lambda execute.
    * @param vin
    *              VIN to find previous primary owners for.
    * @param query
    *              The query to find previous primary owners.
    * @throws IllegalStateException if failed to convert document ID to an
{@link IonValue}.
    */
   public static void previousPrimaryOwners(final TransactionExecutor txn,
final String vin, final String query) {
       try {
           final String docId = VehicleRegistration.getDocumentIdByVin(txn,
vin);
           final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(docId));
           log.info("Querying the 'VehicleRegistration' table's history using
VIN: {}...", vin);
           final Result result = txn.execute(query, parameters);
           ScanTable.printDocuments(result);
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }

   public static void main(final String... args) {
       final String threeMonthsAgo = Instant.now().minus(THREE_MONTHS,
ChronoUnit.DAYS).toString();
       final String query = String.format("SELECT data.Owners.PrimaryOwner,
metadata.version "
                                          + "FROM history(VehicleRegistration,
`%s`) "
                                          + "AS h WHERE h.metadata.id = ?",
threeMonthsAgo);
       ConnectToLedger.getDriver().execute(txn -> {
           final String vin = SampleData.VEHICLES.get(0).getVin();
           previousPrimaryOwners(txn, vin, query);
```

```
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
        log.info("Successfully queried history.");
    }
}
```

> **Note**
>
> - Sie können den Revisionsverlauf eines Dokuments anzeigen, indem Sie die integrierte Verlaufsfunktion in der folgenden Syntax abfragen.
>
>   ```
>   SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
>   [ WHERE h.metadata.id = 'id' ]
>   ```
>
> - Die Start - und Endzeit sind beide optional. Es handelt sich um Amazon Ion-Literalwerte, die mit Backticks (`` `...` ``) gekennzeichnet werden können. Weitere Informationen hierzu finden Sie unter Ion mit PartiQL in Amazon QLDB abfragen.
> - Es hat sich bewährt, eine Verlaufsabfrage sowohl mit einem Datumsbereich (Start- und Endzeit) als auch mit einer Dokument-ID (`metadata.id`) zu qualifizieren. QLDB verarbeitet`SELECT` Abfragen in Transaktionen, für die ein Transaktions-Timeout-Limit gilt.
>
>   Der QLDB-Verlauf wird anhand der Dokument-ID indexiert, und Sie können derzeit keine zusätzlichen Verlaufsindizes erstellen. Verlaufsabfragen, die eine Start- und Endzeit enthalten, profitieren von der Qualifizierung des Datumsbereichs.

2. Kompilieren Sie das`QueryHistory.java` Programm und führen Sie es aus, um den Revisionsverlauf des`VehicleRegistration` Dokuments mit VIN abzufragen`1N4AL11D75C109151`.

Fahren Sie zum kryptografischen Überprüfen einer Dokumentrevision im `vehicle-registration`-Ledger mit Schritt 7: Verifizieren Sie ein Dokument in einem Ledger fort.

Schritt 7: Verifizieren Sie ein Dokument in einem Ledger

Mit Amazon QLDB können Sie die Integrität eines Dokuments im Journal Ihres Hauptbuchs effizient überprüfen, indem Sie kryptografisches Hashing mit SHA-256 verwenden. Weitere

Informationen zur Funktionsweise von Verifizierung und kryptografischem Hashing in QLDB finden Sie unterDatenverifizierung in Amazon QLDB.

In diesem Schritt überprüfen Sie eine Dokumentrevision in der Tabelle `VehicleRegistration` in Ihrem `vehicle-registration`-Ledger. Zuerst fordern Sie einen Digest an, der als Ausgabedatei zurückgegeben wird und als Signatur des gesamten Änderungsverlaufs Ihres Ledgers fungiert. Anschließend fordern Sie einen Nachweis für die Revision in Bezug auf diesen Digest an. Mit diesem Nachweis wird die Integrität Ihrer Revision verifiziert, wenn alle Validierungsprüfungen bestanden werden.

So überprüfen Sie eine Dokumentrevision

1. Sehen Sie sich die folgenden `.java` Dateien an, die QLDB-Objekte darstellen, die für die Überprüfung erforderlich sind, und Hilfsklassen mit Hilfsmethoden für Ion- und Zeichenkettenwerte.

   1. `BlockAddress.java`

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
```

```
package software.amazon.qldb.tutorial.qldb;

import java.util.Objects;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;

/**
 * Represents the BlockAddress field of a QLDB document.
 */
public final class BlockAddress {

    private static final Logger log =
 LoggerFactory.getLogger(BlockAddress.class);

    private final String strandId;
    private final long sequenceNo;

    @JsonCreator
    public BlockAddress(@JsonProperty("strandId") final String strandId,
                        @JsonProperty("sequenceNo") final long sequenceNo) {
        this.strandId = strandId;
        this.sequenceNo = sequenceNo;
    }

    public long getSequenceNo() {
        return sequenceNo;
    }

    public String getStrandId() {
        return strandId;
    }

    @Override
    public String toString() {
        return "BlockAddress{"
                + "strandId='" + strandId + '\''
                + ", sequenceNo=" + sequenceNo
                + '}';
    }
```

```
    @Override
    public boolean equals(final Object o) {
        if (this == o) {
            return true;
        }
        if (o == null || getClass() != o.getClass()) {
            return false;
        }
        BlockAddress that = (BlockAddress) o;
        return sequenceNo == that.sequenceNo
                && strandId.equals(that.strandId);
    }

    @Override
    public int hashCode() {
        // CHECKSTYLE:OFF - Disabling as we are generating a hashCode of multiple
 properties.
        return Objects.hash(strandId, sequenceNo);
        // CHECKSTYLE:ON
    }
}
```

2. Proof.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.amazon.ion.IonReader;
import com.amazon.ion.IonSystem;
import com.amazon.ion.system.IonSystemBuilder;
import com.amazonaws.services.qldb.model.GetRevisionRequest;
import com.amazonaws.services.qldb.model.GetRevisionResult;

import java.util.ArrayList;
import java.util.List;

/**
 * A Java representation of the {@link Proof} object.
 * Returned from the {@link
 com.amazonaws.services.qldb.AmazonQLDB#getRevision(GetRevisionRequest)} api.
 */
public final class Proof {
    private static final IonSystem SYSTEM = IonSystemBuilder.standard().build();

    private List<byte[]> internalHashes;

    public Proof(final List<byte[]> internalHashes) {
        this.internalHashes = internalHashes;
    }

    public List<byte[]> getInternalHashes() {
        return internalHashes;
    }

    /**
     * Decodes a {@link Proof} from an ion text String. This ion text is returned
 in
     * a {@link GetRevisionResult#getProof()}
     *
     * @param ionText
     *              The ion text representing a {@link Proof} object.
     * @return {@link JournalBlock} parsed from the ion text.
```

```
     * @throws IllegalStateException if failed to parse the {@link Proof} object
 from the given ion text.
     */
    public static Proof fromBlob(final String ionText) {
        try {
            IonReader reader = SYSTEM.newReader(ionText);
            List<byte[]> list = new ArrayList<>();
            reader.next();
            reader.stepIn();
            while (reader.next() != null) {
                list.add(reader.newBytes());
            }
            return new Proof(list);
        } catch (Exception e) {
            throw new IllegalStateException("Failed to parse a Proof from byte
 array");
        }
    }
}
```

3. QldbIonUtils.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
```

```
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.amazon.ion.IonReader;
import com.amazon.ion.IonValue;
import com.amazon.ionhash.IonHashReader;
import com.amazon.ionhash.IonHashReaderBuilder;
import com.amazon.ionhash.MessageDigestIonHasherProvider;
import software.amazon.qldb.tutorial.Constants;

public class QldbIonUtils {

    private static MessageDigestIonHasherProvider ionHasherProvider = new
 MessageDigestIonHasherProvider("SHA-256");

    private QldbIonUtils() {}

    /**
     * Builds a hash value from the given {@link IonValue}.
     *
     * @param ionValue
     *              The {@link IonValue} to hash.
     * @return a byte array representing the hash value.
     */
    public static byte[] hashIonValue(final IonValue ionValue) {
        IonReader reader = Constants.SYSTEM.newReader(ionValue);
        IonHashReader hashReader = IonHashReaderBuilder.standard()
                .withHasherProvider(ionHasherProvider)
                .withReader(reader)
                .build();
        while (hashReader.next() != null) {  }
        return hashReader.digest();
    }

}
```

4. QldbStringUtils.java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
```

```
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.amazon.ion.IonWriter;
import com.amazon.ion.system.IonReaderBuilder;
import com.amazon.ion.system.IonTextWriterBuilder;
import com.amazonaws.services.qldb.model.GetBlockResult;
import com.amazonaws.services.qldb.model.GetDigestResult;
import com.amazonaws.services.qldb.model.ValueHolder;

import java.io.IOException;

/**
 * Helper methods to pretty-print certain QLDB response types.
 */
public class QldbStringUtils {

    private QldbStringUtils() {}

    /**
     * Returns the string representation of a given {@link ValueHolder}.
     * Adapted from the AWS SDK autogenerated {@code toString()} method, with
 sensitive values un-redacted.
```

```
     * Additionally, this method pretty-prints any IonText included in the {@link
ValueHolder}.
     *
     * @param valueHolder the {@link ValueHolder} to convert to a String.
     * @return the String representation of the supplied {@link ValueHolder}.
     */
    public static String toUnredactedString(ValueHolder valueHolder) {
        StringBuilder sb = new StringBuilder();
        sb.append("{");
        if (valueHolder.getIonText() != null) {

            sb.append("IonText: ");
            IonWriter prettyWriter = IonTextWriterBuilder.pretty().build(sb);
            try {

prettyWriter.writeValues(IonReaderBuilder.standard().build(valueHolder.getIonText()));
            } catch (IOException ioe) {
                sb.append("**Exception while printing this IonText**");
            }
        }

        sb.append("}");
        return sb.toString();
    }

    /**
     * Returns the string representation of a given {@link GetBlockResult}.
     * Adapted from the AWS SDK autogenerated {@code toString()} method, with
sensitive values un-redacted.
     *
     * @param getBlockResult the {@link GetBlockResult} to convert to a String.
     * @return the String representation of the supplied {@link GetBlockResult}.
     */
    public static String toUnredactedString(GetBlockResult getBlockResult) {
        StringBuilder sb = new StringBuilder();
        sb.append("{");
        if (getBlockResult.getBlock() != null) {
            sb.append("Block:
").append(toUnredactedString(getBlockResult.getBlock())).append(",");
        }

        if (getBlockResult.getProof() != null) {
            sb.append("Proof:
").append(toUnredactedString(getBlockResult.getProof()));
```

```
        }

        sb.append("}");
        return sb.toString();
    }

    /**
     * Returns the string representation of a given {@link GetDigestResult}.
     * Adapted from the AWS SDK autogenerated {@code toString()} method, with
 sensitive values un-redacted.
     *
     * @param getDigestResult the {@link GetDigestResult} to convert to a String.
     * @return the String representation of the supplied {@link GetDigestResult}.
     */
    public static String toUnredactedString(GetDigestResult getDigestResult) {
        StringBuilder sb = new StringBuilder();
        sb.append("{");
        if (getDigestResult.getDigest() != null) {
            sb.append("Digest:
 ").append(getDigestResult.getDigest()).append(",");
        }

        if (getDigestResult.getDigestTipAddress() != null) {
            sb.append("DigestTipAddress:
 ").append(toUnredactedString(getDigestResult.getDigestTipAddress()));
        }

        sb.append("}");
        return sb.toString();
    }
}
```

5. `Verifier.java`

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 copy of this
 * software and associated documentation files (the "Software"), to deal in
 the Software
```

```
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR
 A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Iterator;
import java.util.List;
import java.util.concurrent.ThreadLocalRandom;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazonaws.util.Base64;

import software.amazon.qldb.tutorial.qldb.Proof;

/**
 * Encapsulates the logic to verify the integrity of revisions or blocks in a
 QLDB ledger.
 *
 * The main entry point is {@link #verify(byte[], byte[], String)}.
```

```
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class Verifier {
    public static final Logger log = LoggerFactory.getLogger(Verifier.class);
    private static final int HASH_LENGTH = 32;
    private static final int UPPER_BOUND = 8;

    /**
     * Compares two hashes by their <em>signed</em> byte values in little-
endian order.
     */
    private static Comparator<byte[]> hashComparator = (h1, h2) -> {
        if (h1.length != HASH_LENGTH || h2.length != HASH_LENGTH) {
            throw new IllegalArgumentException("Invalid hash.");
        }
        for (int i = h1.length - 1; i >= 0; i--) {
            int byteEqual = Byte.compare(h1[i], h2[i]);
            if (byteEqual != 0) {
                return byteEqual;
            }
        }

        return 0;
    };

    private Verifier() { }

    /**
     * Verify the integrity of a document with respect to a QLDB ledger
 digest.
     *
     * The verification algorithm includes the following steps:
     *
     * 1. {@link #buildCandidateDigest(Proof, byte[])} build the candidate
 digest from the internal hashes
     * in the {@link Proof}.
     * 2. Check that the {@code candidateLedgerDigest} is equal to the {@code
 ledgerDigest}.
     *
     * @param documentHash
     *              The hash of the document to be verified.
```

```
    * @param digest
    *              The QLDB ledger digest. This digest should have been
retrieved using
    *              {@link com.amazonaws.services.qldb.AmazonQLDB#getDigest}
    * @param proofBlob
    *              The ion encoded bytes representing the {@link Proof}
associated with the supplied
    *              {@code digestTipAddress} and {@code address} retrieved
using
    *              {@link
com.amazonaws.services.qldb.AmazonQLDB#getRevision}.
    * @return {@code true} if the record is verified or {@code false} if it
is not verified.
    */
   public static boolean verify(
           final byte[] documentHash,
           final byte[] digest,
           final String proofBlob
   ) {
       Proof proof = Proof.fromBlob(proofBlob);

       byte[] candidateDigest = buildCandidateDigest(proof, documentHash);

       return Arrays.equals(digest, candidateDigest);
   }

   /**
    * Build the candidate digest representing the entire ledger from the
internal hashes of the {@link Proof}.
    *
    * @param proof
    *              A Java representation of {@link Proof}
    *              returned from {@link
com.amazonaws.services.qldb.AmazonQLDB#getRevision}.
    * @param leafHash
    *              Leaf hash to build the candidate digest with.
    * @return a byte array of the candidate digest.
    */
   private static byte[] buildCandidateDigest(final Proof proof, final byte[]
leafHash) {
       return calculateRootHashFromInternalHashes(proof.getInternalHashes(),
leafHash);
   }
```

```
    /**
     * Get a new instance of {@link MessageDigest} using the SHA-256
algorithm.
     *
     * @return an instance of {@link MessageDigest}.
     * @throws IllegalStateException if the algorithm is not available on the
current JVM.
     */
    static MessageDigest newMessageDigest() {
        try {
            return MessageDigest.getInstance("SHA-256");
        } catch (NoSuchAlgorithmException e) {
            log.error("Failed to create SHA-256 MessageDigest", e);
            throw new IllegalStateException("SHA-256 message digest is
unavailable", e);
        }
    }

    /**
     * Takes two hashes, sorts them, concatenates them, and then returns the
     * hash of the concatenated array.
     *
     * @param h1
     *              Byte array containing one of the hashes to compare.
     * @param h2
     *              Byte array containing one of the hashes to compare.
     * @return the concatenated array of hashes.
     */
    public static byte[] dot(final byte[] h1, final byte[] h2) {
        if (h1.length == 0) {
            return h2;
        }
        if (h2.length == 0) {
            return h1;
        }
        byte[] concatenated = new byte[h1.length + h2.length];
        if (hashComparator.compare(h1, h2) < 0) {
            System.arraycopy(h1, 0, concatenated, 0, h1.length);
            System.arraycopy(h2, 0, concatenated, h1.length, h2.length);
        } else {
            System.arraycopy(h2, 0, concatenated, 0, h2.length);
            System.arraycopy(h1, 0, concatenated, h2.length, h1.length);
        }
        MessageDigest messageDigest = newMessageDigest();
```

```
        messageDigest.update(concatenated);

        return messageDigest.digest();
    }

    /**
     * Starting with the provided {@code leafHash} combined with the provided
{@code internalHashes}
     * pairwise until only the root hash remains.
     *
     * @param internalHashes
     *              Internal hashes of Merkle tree.
     * @param leafHash
     *              Leaf hashes of Merkle tree.
     * @return the root hash.
     */
    private static byte[] calculateRootHashFromInternalHashes(final
List<byte[]> internalHashes, final byte[] leafHash) {
        return internalHashes.stream().reduce(leafHash, Verifier::dot);
    }

    /**
     * Flip a single random bit in the given byte array. This method is used
to demonstrate
     * QLDB's verification features.
     *
     * @param original
     *              The original byte array.
     * @return the altered byte array with a single random bit changed.
     */
    public static byte[] flipRandomBit(final byte[] original) {
        if (original.length == 0) {
            throw new IllegalArgumentException("Array cannot be empty!");
        }
        int alteredPosition =
ThreadLocalRandom.current().nextInt(original.length);
        int b = ThreadLocalRandom.current().nextInt(UPPER_BOUND);
        byte[] altered = new byte[original.length];
        System.arraycopy(original, 0, altered, 0, original.length);
        altered[alteredPosition] = (byte) (altered[alteredPosition] ^ (1 <<
b));
        return altered;
    }
```

```
    public static String toBase64(byte[] arr) {
        return new String(Base64.encode(arr), StandardCharsets.UTF_8);
    }

    /**
     * Convert a {@link ByteBuffer} into byte array.
     *
     * @param buffer
     *              The {@link ByteBuffer} to convert.
     * @return the converted byte array.
     */
    public static byte[] convertByteBufferToByteArray(final ByteBuffer buffer)
{
        byte[] arr = new byte[buffer.remaining()];
        buffer.get(arr);
        return arr;
    }

    /**
     * Calculates the root hash from a list of hashes that represent the base
of a Merkle tree.
     *
     * @param hashes
     *              The list of byte arrays representing hashes making up base
of a Merkle tree.
     * @return a byte array that is the root hash of the given list of hashes.
     */
    public static byte[] calculateMerkleTreeRootHash(List<byte[]> hashes) {
        if (hashes.isEmpty()) {
            return new byte[0];
        }

        List<byte[]> remaining = combineLeafHashes(hashes);
        while (remaining.size() > 1) {
            remaining = combineLeafHashes(remaining);
        }
        return remaining.get(0);
    }

    private static List<byte[]> combineLeafHashes(List<byte[]> hashes) {
        List<byte[]> combinedHashes = new ArrayList<>();
        Iterator<byte[]> it = hashes.stream().iterator();

        while (it.hasNext()) {
```

```
            byte[] left = it.next();
            if (it.hasNext()) {
                byte[] right = it.next();
                byte[] combined = dot(left, right);
                combinedHashes.add(combined);
            } else {
                combinedHashes.add(left);
            }
        }

        return combinedHashes;
    }
}
```

1.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 copy of this
 * software and associated documentation files (the "Software"), to deal in
 the Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR
 A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
```

```
package software.amazon.qldb.tutorial;

import com.amazonaws.util.Base64;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.tutorial.qldb.Proof;

import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.*;
import java.util.concurrent.ThreadLocalRandom;

/**
 * Encapsulates the logic to verify the integrity of revisions or blocks in a
 QLDB ledger.
 *
 * The main entry point is {@link #verify(byte[], byte[], String)}.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class Verifier {
    public static final Logger log = LoggerFactory.getLogger(Verifier.class);
    private static final int HASH_LENGTH = 32;
    private static final int UPPER_BOUND = 8;

    /**
     * Compares two hashes by their <em>signed</em> byte values in little-
endian order.
     */
    private static Comparator<byte[]> hashComparator = (h1, h2) -> {
        if (h1.length != HASH_LENGTH || h2.length != HASH_LENGTH) {
            throw new IllegalArgumentException("Invalid hash.");
        }
        for (int i = h1.length - 1; i >= 0; i--) {
            int byteEqual = Byte.compare(h1[i], h2[i]);
            if (byteEqual != 0) {
                return byteEqual;
            }
        }
```

```
        return 0;
    };

    private Verifier() { }

    /**
     * Verify the integrity of a document with respect to a QLDB ledger
digest.
     *
     * The verification algorithm includes the following steps:
     *
     * 1. {@link #buildCandidateDigest(Proof, byte[])} build the candidate
digest from the internal hashes
     * in the {@link Proof}.
     * 2. Check that the {@code candidateLedgerDigest} is equal to the {@code
ledgerDigest}.
     *
     * @param documentHash
     *              The hash of the document to be verified.
     * @param digest
     *              The QLDB ledger digest. This digest should have been
retrieved using
     *              {@link com.amazonaws.services.qldb.AmazonQLDB#getDigest}
     * @param proofBlob
     *              The ion encoded bytes representing the {@link Proof}
associated with the supplied
     *              {@code digestTipAddress} and {@code address} retrieved
using
     *              {@link
com.amazonaws.services.qldb.AmazonQLDB#getRevision}.
     * @return {@code true} if the record is verified or {@code false} if it
is not verified.
     */
    public static boolean verify(
            final byte[] documentHash,
            final byte[] digest,
            final String proofBlob
    ) {
        Proof proof = Proof.fromBlob(proofBlob);

        byte[] candidateDigest = buildCandidateDigest(proof, documentHash);

        return Arrays.equals(digest, candidateDigest);
    }
```

```
   /**
    * Build the candidate digest representing the entire ledger from the
internal hashes of the {@link Proof}.
    *
    * @param proof
    *              A Java representation of {@link Proof}
    *              returned from {@link
com.amazonaws.services.qldb.AmazonQLDB#getRevision}.
    * @param leafHash
    *              Leaf hash to build the candidate digest with.
    * @return a byte array of the candidate digest.
    */
   private static byte[] buildCandidateDigest(final Proof proof, final byte[]
leafHash) {
       return calculateRootHashFromInternalHashes(proof.getInternalHashes(),
leafHash);
   }

   /**
    * Get a new instance of {@link MessageDigest} using the SHA-256
algorithm.
    *
    * @return an instance of {@link MessageDigest}.
    * @throws IllegalStateException if the algorithm is not available on the
current JVM.
    */
   static MessageDigest newMessageDigest() {
       try {
           return MessageDigest.getInstance("SHA-256");
       } catch (NoSuchAlgorithmException e) {
           log.error("Failed to create SHA-256 MessageDigest", e);
           throw new IllegalStateException("SHA-256 message digest is
unavailable", e);
       }
   }

   /**
    * Takes two hashes, sorts them, concatenates them, and then returns the
    * hash of the concatenated array.
    *
    * @param h1
    *              Byte array containing one of the hashes to compare.
    * @param h2
```

```
     *             Byte array containing one of the hashes to compare.
     * @return the concatenated array of hashes.
     */
    public static byte[] dot(final byte[] h1, final byte[] h2) {
        if (h1.length == 0) {
            return h2;
        }
        if (h2.length == 0) {
            return h1;
        }
        byte[] concatenated = new byte[h1.length + h2.length];
        if (hashComparator.compare(h1, h2) < 0) {
            System.arraycopy(h1, 0, concatenated, 0, h1.length);
            System.arraycopy(h2, 0, concatenated, h1.length, h2.length);
        } else {
            System.arraycopy(h2, 0, concatenated, 0, h2.length);
            System.arraycopy(h1, 0, concatenated, h2.length, h1.length);
        }
        MessageDigest messageDigest = newMessageDigest();
        messageDigest.update(concatenated);

        return messageDigest.digest();
    }

    /**
     * Starting with the provided {@code leafHash} combined with the provided
{@code internalHashes}
     * pairwise until only the root hash remains.
     *
     * @param internalHashes
     *             Internal hashes of Merkle tree.
     * @param leafHash
     *             Leaf hashes of Merkle tree.
     * @return the root hash.
     */
    private static byte[] calculateRootHashFromInternalHashes(final
List<byte[]> internalHashes, final byte[] leafHash) {
        return internalHashes.stream().reduce(leafHash, Verifier::dot);
    }

    /**
     * Flip a single random bit in the given byte array. This method is used
to demonstrate
     * QLDB's verification features.
```

```
     *
     * @param original
     *              The original byte array.
     * @return the altered byte array with a single random bit changed.
     */
    public static byte[] flipRandomBit(final byte[] original) {
        if (original.length == 0) {
            throw new IllegalArgumentException("Array cannot be empty!");
        }
        int alteredPosition =
ThreadLocalRandom.current().nextInt(original.length);
        int b = ThreadLocalRandom.current().nextInt(UPPER_BOUND);
        byte[] altered = new byte[original.length];
        System.arraycopy(original, 0, altered, 0, original.length);
        altered[alteredPosition] = (byte) (altered[alteredPosition] ^ (1 <<
b));
        return altered;
    }

    public static String toBase64(byte[] arr) {
        return new String(Base64.encode(arr), StandardCharsets.UTF_8);
    }

    /**
     * Convert a {@link ByteBuffer} into byte array.
     *
     * @param buffer
     *              The {@link ByteBuffer} to convert.
     * @return the converted byte array.
     */
    public static byte[] convertByteBufferToByteArray(final ByteBuffer buffer)
{
        byte[] arr = new byte[buffer.remaining()];
        buffer.get(arr);
        return arr;
    }

    /**
     * Calculates the root hash from a list of hashes that represent the base
of a Merkle tree.
     *
     * @param hashes
     *              The list of byte arrays representing hashes making up base
of a Merkle tree.
```

```
     * @return a byte array that is the root hash of the given list of hashes.
     */
    public static byte[] calculateMerkleTreeRootHash(List<byte[]> hashes) {
        if (hashes.isEmpty()) {
            return new byte[0];
        }

        List<byte[]> remaining = combineLeafHashes(hashes);
        while (remaining.size() > 1) {
            remaining = combineLeafHashes(remaining);
        }
        return remaining.get(0);
    }

    private static List<byte[]> combineLeafHashes(List<byte[]> hashes) {
        List<byte[]> combinedHashes = new ArrayList<>();
        Iterator<byte[]> it = hashes.stream().iterator();

        while (it.hasNext()) {
            byte[] left = it.next();
            if (it.hasNext()) {
                byte[] right = it.next();
                byte[] combined = dot(left, right);
                combinedHashes.add(combined);
            } else {
                combinedHashes.add(left);
            }
        }

        return combinedHashes;
    }
}
```

2. Verwenden Sie zwei `.java` Dateien (`GetDigest.javaundGetRevision.java`), um die folgenden Schritte durchzuführen:

   - Fordern Sie einen neuen Digest aus dem Ledger `vehicle-registration` an.
   - Fordern Sie einen Nachweis für jede Revision eines Dokuments aus der `VehicleRegistration`-Tabelle an.
   - Überprüfen Sie die Revisionen mit dem zurückgegebenen Digest und dem Nachweis, indem Sie den Digest neu berechnen.

Das Programm `GetDigest.java` enthält den folgenden Code.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.model.GetDigestRequest;
import com.amazonaws.services.qldb.model.GetDigestResult;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.tutorial.qldb.QldbStringUtils;

/**
 * This is an example for retrieving the digest of a particular ledger.
 *
 * This code expects that you have AWS credentials setup per:
```

```
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class GetDigest {
    public static final Logger log = LoggerFactory.getLogger(GetDigest.class);
    public static AmazonQLDB client = CreateLedger.getClient();

    private GetDigest() { }

    /**
     * Calls {@link #getDigest(String)} for a ledger.
     *
     * @param args
     *              Arbitrary command-line arguments.
     * @throws Exception if failed to get a ledger digest.
     */
    public static void main(final String... args) throws Exception {
        try {

            getDigest(Constants.LEDGER_NAME);

        } catch (Exception e) {
            log.error("Unable to get a ledger digest!", e);
            throw e;
        }
    }

    /**
     * Get the digest for the specified ledger.
     *
     * @param ledgerName
     *              The ledger to get digest from.
     * @return {@link GetDigestResult}.
     */
    public static GetDigestResult getDigest(final String ledgerName) {
        log.info("Let's get the current digest of the ledger named {}.",
 ledgerName);
        GetDigestRequest request = new GetDigestRequest()
                .withName(ledgerName);
        GetDigestResult result = client.getDigest(request);
        log.info("Success. LedgerDigest: {}.",
 QldbStringUtils.toUnredactedString(result));
        return result;
    }
```

```
}
```

> **Note**
>
> Verwenden Sie die `getDigest`-Methode, um einen Digest anzufordern, der die aktuelle Spitze des Journal in Ihrem Ledger abdeckt. Die Notiz des Journals bezieht sich auf den letzten übernommenen Block zu dem Zeitpunkt, zu dem QLDB Ihre Anfrage erhält.

Das Programm `GetRevision.java` enthält den folgenden Code.

2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonReader;
```

```
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.IonWriter;
import com.amazon.ion.system.IonReaderBuilder;
import com.amazon.ion.system.IonSystemBuilder;
import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.model.GetDigestResult;
import com.amazonaws.services.qldb.model.GetRevisionRequest;
import com.amazonaws.services.qldb.model.GetRevisionResult;
import com.amazonaws.services.qldb.model.ValueHolder;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.QldbDriver;
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.qldb.BlockAddress;
import software.amazon.qldb.tutorial.qldb.QldbRevision;
import software.amazon.qldb.tutorial.qldb.QldbStringUtils;

/**
 * Verify the integrity of a document revision in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class GetRevision {
    public static final Logger log = LoggerFactory.getLogger(GetRevision.class);
    public static AmazonQLDB client = CreateLedger.getClient();
    private static final IonSystem SYSTEM = IonSystemBuilder.standard().build();

    private GetRevision() { }

    public static void main(String... args) throws Exception {

        final String vin = SampleData.REGISTRATIONS.get(0).getVin();
```

```
        verifyRegistration(ConnectToLedger.getDriver(), Constants.LEDGER_NAME,
 vin);
    }

    /**
     * Verify each version of the registration for the given VIN.
     *
     * @param driver
     *              A QLDB driver.
     * @param ledgerName
     *              The ledger to get digest from.
     * @param vin
     *              VIN to query the revision history of a specific registration
 with.
     * @throws Exception if failed to verify digests.
     * @throws AssertionError if document revision verification failed.
     */
    public static void verifyRegistration(final QldbDriver driver, final String
 ledgerName, final String vin)
            throws Exception {
        log.info(String.format("Let's verify the registration with VIN=%s, in
 ledger=%s.", vin, ledgerName));

        try {
            log.info("First, let's get a digest.");
            GetDigestResult digestResult = GetDigest.getDigest(ledgerName);

            ValueHolder digestTipAddress = digestResult.getDigestTipAddress();
            byte[] digestBytes =
 Verifier.convertByteBufferToByteArray(digestResult.getDigest());

            log.info("Got a ledger digest. Digest end address={}, digest={}.",
                QldbStringUtils.toUnredactedString(digestTipAddress),
                Verifier.toBase64(digestBytes));

            log.info(String.format("Next, let's query the registration with VIN=
%s. "
                    + "Then we can verify each version of the registration.",
 vin));
            List<IonStruct> documentsWithMetadataList = new ArrayList<>();
            driver.execute(txn -> {
                documentsWithMetadataList.addAll(queryRegistrationsByVin(txn,
 vin));
```

```
            });
            log.info("Registrations queried successfully!");

            log.info(String.format("Found %s revisions of the registration with
VIN=%s.",
                    documentsWithMetadataList.size(), vin));

            for (IonStruct ionStruct : documentsWithMetadataList) {

                QldbRevision document = QldbRevision.fromIon(ionStruct);
                log.info(String.format("Let's verify the document: %s",
document));

                log.info("Let's get a proof for the document.");
                GetRevisionResult proofResult = getRevision(
                        ledgerName,
                        document.getMetadata().getId(),
                        digestTipAddress,
                        document.getBlockAddress()
                );

                final IonValue proof =
Constants.MAPPER.writeValueAsIonValue(proofResult.getProof());
                final IonReader reader =
IonReaderBuilder.standard().build(proof);
                reader.next();
                ByteArrayOutputStream baos = new ByteArrayOutputStream();
                IonWriter writer = SYSTEM.newBinaryWriter(baos);
                writer.writeValue(reader);
                writer.close();
                baos.flush();
                baos.close();
                byte[] byteProof = baos.toByteArray();

                log.info(String.format("Got back a proof: %s",
Verifier.toBase64(byteProof)));

                boolean verified = Verifier.verify(
                        document.getHash(),
                        digestBytes,
                        proofResult.getProof().getIonText()
                );

                if (!verified) {
```

```
                    throw new AssertionError("Document revision is not
verified!");
                } else {
                    log.info("Success! The document is verified");
                }

                byte[] alteredDigest = Verifier.flipRandomBit(digestBytes);
                log.info(String.format("Flipping one bit in the digest and
assert that the document is NOT verified. "
                        + "The altered digest is: %s",
Verifier.toBase64(alteredDigest)));
                verified = Verifier.verify(
                        document.getHash(),
                        alteredDigest,
                        proofResult.getProof().getIonText()
                );

                if (verified) {
                    throw new AssertionError("Expected document to not be
verified against altered digest.");
                } else {
                    log.info("Success! As expected flipping a bit in the digest
causes verification to fail.");
                }

                byte[] alteredDocumentHash =
Verifier.flipRandomBit(document.getHash());
                log.info(String.format("Flipping one bit in the document's hash
and assert that it is NOT verified. "
                        + "The altered document hash is: %s.",
Verifier.toBase64(alteredDocumentHash)));
                verified = Verifier.verify(
                        alteredDocumentHash,
                        digestBytes,
                        proofResult.getProof().getIonText()
                );

                if (verified) {
                    throw new AssertionError("Expected altered document hash to
not be verified against digest.");
                } else {
                    log.info("Success! As expected flipping a bit in the
document hash causes verification to fail.");
                }
```

```
            }

        } catch (Exception e) {
            log.error("Failed to verify digests.", e);
            throw e;
        }

        log.info(String.format("Finished verifying the registration with VIN=%s
in ledger=%s.", vin, ledgerName));
    }

    /**
     * Get the revision of a particular document specified by the given document
ID and block address.
     *
     * @param ledgerName
     *              Name of the ledger containing the document.
     * @param documentId
     *              Unique ID for the document to be verified, contained in the
committed view of the document.
     * @param digestTipAddress
     *              The latest block location covered by the digest.
     * @param blockAddress
     *              The location of the block to request.
     * @return the requested revision.
     */
    public static GetRevisionResult getRevision(final String ledgerName, final
String documentId,
                                                final ValueHolder
digestTipAddress, final BlockAddress blockAddress) {
        try {
            GetRevisionRequest request = new GetRevisionRequest()
                    .withName(ledgerName)
                    .withDigestTipAddress(digestTipAddress)
                    .withBlockAddress(new
ValueHolder().withIonText(Constants.MAPPER.writeValueAsIonValue(blockAddress)
                            .toString()))
                    .withDocumentId(documentId);
            return client.getRevision(request);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }
```

```
    /**
     * Query the registration history for the given VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              The unique VIN to query.
     * @return a list of {@link IonStruct} representing the registration
 history.
     * @throws IllegalStateException if failed to convert parameters into {@link
 IonValue}
     */
    public static List<IonStruct> queryRegistrationsByVin(final
 TransactionExecutor txn, final String vin) {
        log.info(String.format("Let's query the 'VehicleRegistration' table for
 VIN: %s...", vin));
        log.info("Let's query the 'VehicleRegistration' table for VIN: {}...",
 vin);
        final String query = String.format("SELECT * FROM _ql_committed_%s WHERE
 data.VIN = ?",
                Constants.VEHICLE_REGISTRATION_TABLE_NAME);
        try {
            final List<IonValue> parameters =
 Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
            final Result result = txn.execute(query, parameters);
            List<IonStruct> list = ScanTable.toIonStructs(result);
            log.info(String.format("Found %d document(s)!", list.size()));
            return list;
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }
}
```

1.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
```

```
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonReader;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonValue;
import com.amazon.ion.IonWriter;
import com.amazon.ion.system.IonReaderBuilder;
import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.model.GetDigestResult;
import com.amazonaws.services.qldb.model.GetRevisionRequest;
import com.amazonaws.services.qldb.model.GetRevisionResult;
import com.amazonaws.services.qldb.model.ValueHolder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.QldbSession;
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.qldb.BlockAddress;
import software.amazon.qldb.tutorial.qldb.QldbRevision;
import software.amazon.qldb.tutorial.qldb.QldbStringUtils;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
```

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Verify the integrity of a document revision in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class GetRevision {
    public static final Logger log = LoggerFactory.getLogger(GetRevision.class);
    public static AmazonQLDB client = CreateLedger.getClient();

    private GetRevision() { }

    public static void main(String... args) throws Exception {

        final String vin = SampleData.REGISTRATIONS.get(0).getVin();

        try (QldbSession qldbSession = ConnectToLedger.createQldbSession()) {
            verifyRegistration(qldbSession, Constants.LEDGER_NAME, vin);
        }
    }

    /**
     * Verify each version of the registration for the given VIN.
     *
     * @param qldbSession
     *              A QLDB session.
     * @param ledgerName
     *              The ledger to get digest from.
     * @param vin
     *              VIN to query the revision history of a specific registration
 with.
     * @throws Exception if failed to verify digests.
     * @throws AssertionError if document revision verification failed.
     */
    public static void verifyRegistration(final QldbSession qldbSession, final
 String ledgerName, final String vin)
            throws Exception {
        log.info(String.format("Let's verify the registration with VIN=%s, in
 ledger=%s.", vin, ledgerName));
```

```
        try {
            log.info("First, let's get a digest.");
            GetDigestResult digestResult = GetDigest.getDigest(ledgerName);

            ValueHolder digestTipAddress = digestResult.getDigestTipAddress();
            byte[] digestBytes =
 Verifier.convertByteBufferToByteArray(digestResult.getDigest());

            log.info("Got a ledger digest. Digest end address={}, digest={}.",
                QldbStringUtils.toUnredactedString(digestTipAddress),
                Verifier.toBase64(digestBytes));

            log.info(String.format("Next, let's query the registration with VIN=
%s. "
                    + "Then we can verify each version of the registration.",
 vin));
            List<IonStruct> documentsWithMetadataList = new ArrayList<>();
            qldbSession.execute(txn -> {
                documentsWithMetadataList.addAll(queryRegistrationsByVin(txn,
 vin));
            }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
            log.info("Registrations queried successfully!");

            log.info(String.format("Found %s revisions of the registration with
 VIN=%s.",
                    documentsWithMetadataList.size(), vin));

            for (IonStruct ionStruct : documentsWithMetadataList) {

                QldbRevision document = QldbRevision.fromIon(ionStruct);
                log.info(String.format("Let's verify the document: %s",
 document));

                log.info("Let's get a proof for the document.");
                GetRevisionResult proofResult = getRevision(
                        ledgerName,
                        document.getMetadata().getId(),
                        digestTipAddress,
                        document.getBlockAddress()
                );

                final IonValue proof =
 Constants.MAPPER.writeValueAsIonValue(proofResult.getProof());
```

```
                final IonReader reader =
IonReaderBuilder.standard().build(proof);
                reader.next();
                ByteArrayOutputStream baos = new ByteArrayOutputStream();
                IonWriter writer = Constants.SYSTEM.newBinaryWriter(baos);
                writer.writeValue(reader);
                writer.close();
                baos.flush();
                baos.close();
                byte[] byteProof = baos.toByteArray();

                log.info(String.format("Got back a proof: %s",
Verifier.toBase64(byteProof)));

                boolean verified = Verifier.verify(
                        document.getHash(),
                        digestBytes,
                        proofResult.getProof().getIonText()
                );

                if (!verified) {
                    throw new AssertionError("Document revision is not
verified!");
                } else {
                    log.info("Success! The document is verified");
                }

                byte[] alteredDigest = Verifier.flipRandomBit(digestBytes);
                log.info(String.format("Flipping one bit in the digest and
assert that the document is NOT verified. "
                        + "The altered digest is: %s",
Verifier.toBase64(alteredDigest)));
                verified = Verifier.verify(
                        document.getHash(),
                        alteredDigest,
                        proofResult.getProof().getIonText()
                );

                if (verified) {
                    throw new AssertionError("Expected document to not be
verified against altered digest.");
                } else {
                    log.info("Success! As expected flipping a bit in the digest
causes verification to fail.");
```

```
                }

                byte[] alteredDocumentHash =
Verifier.flipRandomBit(document.getHash());
                log.info(String.format("Flipping one bit in the document's hash
and assert that it is NOT verified. "
                        + "The altered document hash is: %s.",
Verifier.toBase64(alteredDocumentHash)));
                verified = Verifier.verify(
                        alteredDocumentHash,
                        digestBytes,
                        proofResult.getProof().getIonText()
                );

                if (verified) {
                    throw new AssertionError("Expected altered document hash to
not be verified against digest.");
                } else {
                    log.info("Success! As expected flipping a bit in the
document hash causes verification to fail.");
                }
            }

        } catch (Exception e) {
            log.error("Failed to verify digests.", e);
            throw e;
        }

        log.info(String.format("Finished verifying the registration with VIN=%s
in ledger=%s.", vin, ledgerName));
    }

    /**
     * Get the revision of a particular document specified by the given document
ID and block address.
     *
     * @param ledgerName
     *              Name of the ledger containing the document.
     * @param documentId
     *              Unique ID for the document to be verified, contained in the
committed view of the document.
     * @param digestTipAddress
     *              The latest block location covered by the digest.
     * @param blockAddress
```

```
     *              The location of the block to request.
     * @return the requested revision.
     */
    public static GetRevisionResult getRevision(final String ledgerName, final
String documentId,
                                                final ValueHolder
digestTipAddress, final BlockAddress blockAddress) {
        try {
            GetRevisionRequest request = new GetRevisionRequest()
                    .withName(ledgerName)
                    .withDigestTipAddress(digestTipAddress)
                    .withBlockAddress(new
ValueHolder().withIonText(Constants.MAPPER.writeValueAsIonValue(blockAddress)
                            .toString()))
                    .withDocumentId(documentId);
            return client.getRevision(request);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Query the registration history for the given VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              The unique VIN to query.
     * @return a list of {@link IonStruct} representing the registration
history.
     * @throws IllegalStateException if failed to convert parameters into {@link
IonValue}
     */
    public static List<IonStruct> queryRegistrationsByVin(final
TransactionExecutor txn, final String vin) {
        log.info(String.format("Let's query the 'VehicleRegistration' table for
VIN: %s...", vin));
        log.info("Let's query the 'VehicleRegistration' table for VIN: {}...",
vin);
        final String query = String.format("SELECT * FROM _ql_committed_%s WHERE
data.VIN = ?",
                Constants.VEHICLE_REGISTRATION_TABLE_NAME);
        try {
```

```
            final List<IonValue> parameters =
 Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
            final Result result = txn.execute(query, parameters);
            List<IonStruct> list = ScanTable.toIonStructs(result);
            log.info(String.format("Found %d document(s)!", list.size()));
            return list;
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }
}
```

> **Note**
>
> Nachdem die `getRevision`-Methode einen Nachweis für die angegebene Dokumentrevision zurückgibt, verwendet dieses Programm eine clientseitige API, um diese Revision zu überprüfen. Eine Übersicht über den Algorithmus, der von dieser API verwendet wird, finden Sie unter Verwenden eines Beweises zur Neuberechnung Ihres Digest.

3. Kompilieren Sie das `GetRevision.java`-Programm und führen Sie es aus, um das `VehicleRegistration`-Dokument kryptografisch mit VIN `1N4AL11D75C109151` zu überprüfen.

Um Journaldaten im`vehicle-registration` Ledger zu exportieren und zu validieren, fahren Sie mit fortSchritt 8: Exportieren und validieren Sie Journaldaten in einem Ledger.

Schritt 8: Exportieren und validieren Sie Journaldaten in einem Ledger

In Amazon QLDB können Sie für verschiedene Zwecke wie Datenspeicherung, Analyse und Prüfung auf den Inhalt des Journals in Ihrem Hauptbuch zugreifen. Weitere Informationen finden Sie unter Exportieren von Journaldaten aus Amazon QLDB.

In diesem Schritt exportieren Sie Journalblöcke aus dem`vehicle-registration` Ledger in einen Amazon S3 S3-Bucket. Anschließend verwenden Sie die exportierten Daten, um die Hash-Kette zwischen Journalblöcken und den einzelnen Hash-Komponenten innerhalb jedes Blocks zu validieren.

DieAWS Identity and Access Management (IAM) -Prinzipalentität, die Sie verwenden, muss über ausreichende IAM-Berechtigungen verfügen, um einen Amazon S3 S3-Bucket in Ihrem zu erstellenAWS-Konto. Informationen finden Sie unter Richtlinien und Berechtigungen in Amazon S3 im Amazon S3 S3-Benutzerhandbuch. Sie benötigen außerdem die Berechtigungen, um eine IAM-Rolle mit einer angehängten Berechtigungsrichtlinie zu erstellen, die es QLDB ermöglicht, Objekte in Ihren Amazon S3 S3-Bucket zu schreiben. Weitere Informationen finden Sie unter Erforderliche Berechtigungen für den Zugriff auf IAM-Ressourcen im IAM-Benutzerhandbuch.

Um Journaldaten zu exportieren und zu validieren

1. Überprüfen Sie die folgende Datei (`JournalBlock.java`), die einen Journalblock und seinen Dateninhalt darstellt. Es enthält eine benannte Methode`verifyBlockHash()`, die demonstriert, wie jede einzelne Komponente eines Block-Hashs berechnet wird.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.fasterxml.jackson.annotation.JsonCreator;
```

```
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import com.fasterxml.jackson.dataformat.ion.IonTimestampSerializers;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.Verifier;

import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Arrays;
import java.util.Date;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

import static java.nio.ByteBuffer.wrap;

/**
 * Represents a JournalBlock that was recorded after executing a transaction
 * in the ledger.
 */
public final class JournalBlock {
    private static final Logger log = LoggerFactory.getLogger(JournalBlock.class);

    private BlockAddress blockAddress;
    private String transactionId;
    @JsonSerialize(using =
 IonTimestampSerializers.IonTimestampJavaDateSerializer.class)
    private Date blockTimestamp;
    private byte[] blockHash;
    private byte[] entriesHash;
    private byte[] previousBlockHash;
    private byte[][] entriesHashList;
    private TransactionInfo transactionInfo;
    private RedactionInfo redactionInfo;
    private List<QldbRevision> revisions;

    @JsonCreator
    public JournalBlock(@JsonProperty("blockAddress") final BlockAddress
 blockAddress,
                        @JsonProperty("transactionId") final String transactionId,
                        @JsonProperty("blockTimestamp") final Date blockTimestamp,
```

```
                        @JsonProperty("blockHash") final byte[] blockHash,
                        @JsonProperty("entriesHash") final byte[] entriesHash,
                        @JsonProperty("previousBlockHash") final byte[]
previousBlockHash,
                        @JsonProperty("entriesHashList") final byte[][]
entriesHashList,
                        @JsonProperty("transactionInfo") final TransactionInfo
transactionInfo,
                        @JsonProperty("redactionInfo") final RedactionInfo
redactionInfo,
                        @JsonProperty("revisions") final List<QldbRevision>
revisions) {
       this.blockAddress = blockAddress;
       this.transactionId = transactionId;
       this.blockTimestamp = blockTimestamp;
       this.blockHash = blockHash;
       this.entriesHash = entriesHash;
       this.previousBlockHash = previousBlockHash;
       this.entriesHashList = entriesHashList;
       this.transactionInfo = transactionInfo;
       this.redactionInfo = redactionInfo;
       this.revisions = revisions;
   }

   public BlockAddress getBlockAddress() {
       return blockAddress;
   }

   public String getTransactionId() {
       return transactionId;
   }

   public Date getBlockTimestamp() {
       return blockTimestamp;
   }

   public byte[][] getEntriesHashList() {
       return entriesHashList;
   }

   public TransactionInfo getTransactionInfo() {
       return transactionInfo;
   }
```

```
    public RedactionInfo getRedactionInfo() {
        return redactionInfo;
    }

    public List<QldbRevision> getRevisions() {
        return revisions;
    }

    public byte[] getEntriesHash() {
        return entriesHash;
    }

    public byte[] getBlockHash() {
        return blockHash;
    }

    public byte[] getPreviousBlockHash() {
        return previousBlockHash;
    }

    @Override
    public String toString() {
        return "JournalBlock{"
                + "blockAddress=" + blockAddress
                + ", transactionId='" + transactionId + '\''
                + ", blockTimestamp=" + blockTimestamp
                + ", blockHash=" + Arrays.toString(blockHash)
                + ", entriesHash=" + Arrays.toString(entriesHash)
                + ", previousBlockHash=" + Arrays.toString(previousBlockHash)
                + ", entriesHashList=" + Arrays.toString(entriesHashList)
                + ", transactionInfo=" + transactionInfo
                + ", redactionInfo=" + redactionInfo
                + ", revisions=" + revisions
                + '}';
    }

    @Override
    public boolean equals(final Object o) {
        if (this == o) {
            return true;
        }
        if (!(o instanceof JournalBlock)) {
            return false;
        }
```

```
        final JournalBlock that = (JournalBlock) o;

        if (!getBlockAddress().equals(that.getBlockAddress())) {
            return false;
        }
        if (!getTransactionId().equals(that.getTransactionId())) {
            return false;
        }
        if (!getBlockTimestamp().equals(that.getBlockTimestamp())) {
            return false;
        }
        if (!Arrays.equals(getBlockHash(), that.getBlockHash())) {
            return false;
        }
        if (!Arrays.equals(getEntriesHash(), that.getEntriesHash())) {
            return false;
        }
        if (!Arrays.equals(getPreviousBlockHash(), that.getPreviousBlockHash())) {
            return false;
        }
        if (!Arrays.deepEquals(getEntriesHashList(), that.getEntriesHashList())) {
            return false;
        }
        if (!getTransactionInfo().equals(that.getTransactionInfo())) {
            return false;
        }
        if (getRedactionInfo() != null ? !
getRedactionInfo().equals(that.getRedactionInfo()) : that.getRedactionInfo() !=
 null) {
            return false;
        }
        return getRevisions() != null ?
 getRevisions().equals(that.getRevisions()) : that.getRevisions() == null;
    }

    @Override
    public int hashCode() {
        int result = getBlockAddress().hashCode();
        result = 31 * result + getTransactionId().hashCode();
        result = 31 * result + getBlockTimestamp().hashCode();
        result = 31 * result + Arrays.hashCode(getBlockHash());
        result = 31 * result + Arrays.hashCode(getEntriesHash());
        result = 31 * result + Arrays.hashCode(getPreviousBlockHash());
```

```
        result = 31 * result + Arrays.deepHashCode(getEntriesHashList());
        result = 31 * result + getTransactionInfo().hashCode();
        result = 31 * result + (getRedactionInfo() != null ?
getRedactionInfo().hashCode() : 0);
        result = 31 * result + (getRevisions() != null ?
getRevisions().hashCode() : 0);
        return result;
    }

    /**
     * This method validates that the hashes of the components of a journal block
make up the block
     * hash that is provided with the block itself.
     *
     * The components that contribute to the hash of the journal block consist of
the following:
     *   - user transaction information (contained in [transactionInfo])
     *   - user redaction information (contained in [redactionInfo])
     *   - user revisions (contained in [revisions])
     *   - hashes of internal-only system metadata (contained in [revisions] and in
[entriesHashList])
     *   - the previous block hash
     *
     * If any of the computed hashes of user information cannot be validated or any
of the system
     * hashes do not result in the correct computed values, this method will throw
an IllegalArgumentException.
     *
     * Internal-only system metadata is represented by its hash, and can be present
in the form of certain
     * items in the [revisions] list that only contain a hash and no user data, as
well as some hashes
     * in [entriesHashList].
     *
     * To validate that the hashes of the user data are valid components of the
[blockHash], this method
     * performs the following steps:
     *
     * 1. Compute the hash of the [transactionInfo] and validate that it is
included in the [entriesHashList].
     * 2. Compute the hash of the [redactionInfo], if present, and validate that it
is included in the [entriesHashList].
     * 3. Validate the hash of each user revision was correctly computed and
matches the hash published
```

```
     * with that revision.
     * 4. Compute the hash of the [revisions] by treating the revision hashes as
 the leaf nodes of a Merkle tree
     * and calculating the root hash of that tree. Then validate that hash is
 included in the [entriesHashList].
     * 5. Compute the hash of the [entriesHashList] by treating the hashes as the
 leaf nodes of a Merkle tree
     * and calculating the root hash of that tree. Then validate that hash matches
 [entriesHash].
     * 6. Finally, compute the block hash by computing the hash resulting from
 concatenating the [entriesHash]
     * and previous block hash, and validate that the result matches the
 [blockHash] provided by QLDB with the block.
     *
     * This method is called by ValidateQldbHashChain::verify for each journal
 block to validate its
     * contents before verifying that the hash chain between consecutive blocks is
 correct.
     */
    public void verifyBlockHash() {
        Set<ByteBuffer> entriesHashSet = new HashSet<>();
        Arrays.stream(entriesHashList).forEach(hash ->
 entriesHashSet.add(wrap(hash).asReadOnlyBuffer()));

        byte[] computedTransactionInfoHash = computeTransactionInfoHash();
        if (!
entriesHashSet.contains(wrap(computedTransactionInfoHash).asReadOnlyBuffer())) {
            throw new IllegalArgumentException(
                    "Block transactionInfo hash is not contained in the QLDB block
 entries hash list.");
        }

        if (redactionInfo != null) {
            byte[] computedRedactionInfoHash = computeRedactionInfoHash();
            if (!
entriesHashSet.contains(wrap(computedRedactionInfoHash).asReadOnlyBuffer())) {
                throw new IllegalArgumentException(
                        "Block redactionInfo hash is not contained in the QLDB
 block entries hash list.");
            }
        }

        if (revisions != null) {
            revisions.forEach(QldbRevision::verifyRevisionHash);
```

```
            byte[] computedRevisionsHash = computeRevisionsHash();
            if (!
entriesHashSet.contains(wrap(computedRevisionsHash).asReadOnlyBuffer())) {
                throw new IllegalArgumentException(
                        "Block revisions list hash is not contained in the QLDB
 block entries hash list.");
            }
        }

        byte[] computedEntriesHash = computeEntriesHash();
        if (!Arrays.equals(computedEntriesHash, entriesHash)) {
            throw new IllegalArgumentException("Computed entries hash does not
 match entries hash provided in the block.");
        }

        byte[] computedBlockHash = Verifier.dot(computedEntriesHash,
 previousBlockHash);
        if (!Arrays.equals(computedBlockHash, blockHash)) {
            throw new IllegalArgumentException("Computed block hash does not match
 block hash provided in the block.");
        }
    }

    private byte[] computeTransactionInfoHash() {
        try {
            return
 QldbIonUtils.hashIonValue(Constants.MAPPER.writeValueAsIonValue(transactionInfo));
        } catch (IOException e) {
            throw new IllegalArgumentException("Could not compute transactionInfo
 hash to verify block hash.", e);
        }
    }

    private byte[] computeRedactionInfoHash() {
        try {
            return
 QldbIonUtils.hashIonValue(Constants.MAPPER.writeValueAsIonValue(redactionInfo));
        } catch (IOException e) {
            throw new IllegalArgumentException("Could not compute redactionInfo
 hash to verify block hash.", e);
        }
    }

    private byte[] computeRevisionsHash() {
```

```
        return
 Verifier.calculateMerkleTreeRootHash(revisions.stream().map(QldbRevision::getHash).collect
    }

    private byte[] computeEntriesHash() {
        return
 Verifier.calculateMerkleTreeRootHash(Arrays.asList(entriesHashList));
    }
}
```

2. Kompilieren Sie das folgende Programm (`ValidateQldbHashChain.java`) und führen Sie es aus, um die folgenden Schritte auszuführen:

   1. Exportieren Sie Journalblöcke aus dem`vehicle-registration` Ledger in einen Amazon S3 S3-Bucket mit dem Namen**`qldb-tutorial-journal-export-`*`111122223333`*** (ersetzen Sie ihn durch IhreAWS-Konto Nummer).
   2. Validieren Sie die einzelnen Hash-Komponenten in jedem Block, indem Sie aufrufen`verifyBlockHash()`.
   3. Validieren Sie die Hash-Kette zwischen Journalblöcken.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
```

```
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.services.qldb.model.ExportJournalToS3Result;
import com.amazonaws.services.qldb.model.S3EncryptionConfiguration;
import com.amazonaws.services.qldb.model.S3ObjectEncryptionType;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

import java.time.Instant;
import java.util.Arrays;
import java.util.List;

import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.GetCallerIdentityRequest;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.tutorial.qldb.JournalBlock;

/**
 * Validate the hash chain of a QLDB ledger by stepping through its S3 export.
 *
 * This code accepts an exportId as an argument, if exportId is passed the code
 * will use that or request QLDB to generate a new export to perform QLDB hash
 * chain validation.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class ValidateQldbHashChain {
    public static final Logger log =
 LoggerFactory.getLogger(ValidateQldbHashChain.class);
    private static final int TIME_SKEW = 20;

    private ValidateQldbHashChain() { }

    /**
     * Export journal contents to a S3 bucket.
     *
     * @return the ExportId of the journal export.
```

```
     * @throws InterruptedException if the thread is interrupted while waiting for
export to complete.
     */
    private static String createExport() throws InterruptedException {
        String accountId = AWSSecurityTokenServiceClientBuilder.defaultClient()
            .getCallerIdentity(new GetCallerIdentityRequest()).getAccount();
        String bucketName = Constants.JOURNAL_EXPORT_S3_BUCKET_NAME_PREFIX + "-" +
accountId;
        String prefix = Constants.LEDGER_NAME + "-" +
Instant.now().getEpochSecond() + "/";

        S3EncryptionConfiguration encryptionConfiguration = new
S3EncryptionConfiguration()
                .withObjectEncryptionType(S3ObjectEncryptionType.SSE_S3);
        ExportJournalToS3Result exportJournalToS3Result =

ExportJournal.createJournalExportAndAwaitCompletion(Constants.LEDGER_NAME,
                    bucketName, prefix, null, encryptionConfiguration,
ExportJournal.DEFAULT_EXPORT_TIMEOUT_MS);

        return exportJournalToS3Result.getExportId();
    }

    /**
     * Validates that the chain hash on the {@link JournalBlock} is valid.
     *
     * @param journalBlocks
     *              {@link JournalBlock} containing hashes to validate.
     * @throws IllegalStateException if previous block hash does not match.
     */
    public static void verify(final List<JournalBlock> journalBlocks) {
        if (journalBlocks.size() == 0) {
            return;
        }

        journalBlocks.stream().reduce(null, (previousJournalBlock, journalBlock) ->
{
            journalBlock.verifyBlockHash();
            if (previousJournalBlock == null) { return journalBlock; }
            if (!Arrays.equals(previousJournalBlock.getBlockHash(),
journalBlock.getPreviousBlockHash())) {
                throw new IllegalStateException("Previous block hash doesn't
match.");
            }
```

```
            byte[] blockHash = Verifier.dot(journalBlock.getEntriesHash(),
 previousJournalBlock.getBlockHash());
            if (!Arrays.equals(blockHash, journalBlock.getBlockHash())) {
                throw new IllegalStateException("Block hash doesn't match
 entriesHash dot previousBlockHash, the chain is "
                        + "broken.");
            }
            return journalBlock;
        });
    }

    public static void main(final String... args) throws InterruptedException {
        try {
            String exportId;
            if (args.length == 1) {
                exportId = args[0];
                log.info("Validating QLDB hash chain for exportId: " + exportId);
            } else {
                log.info("Requesting QLDB to create an export.");
                exportId = createExport();
            }
            List<JournalBlock> journalBlocks =

 JournalS3ExportReader.readExport(DescribeJournalExport.describeExport(Constants.LEDGER_NAM
                    exportId), AmazonS3ClientBuilder.defaultClient());
            verify(journalBlocks);
        } catch (Exception e) {
            log.error("Unable to perform hash chain verification.", e);
            throw e;
        }
    }

}
```

Wenn Sie den `vehicle-registration`-Ledger nicht mehr verwenden müssen, fahren Sie mit Schritt 9 (optional): Bereinigen von Ressourcen fort.

Schritt 9 (optional): Bereinigen von Ressourcen

Sie können den `vehicle-registration`-Ledger weiterhin verwenden. Wenn Sie ihn nicht mehr benötigen, sollten Sie ihn jedoch löschen.

So löschen Sie den Ledger

1. Kompilieren Sie das folgende Programm (`DeleteLedger.java`) und führen Sie es aus, um Ihren `vehicle-registration`-Ledger und seinen gesamten Inhalt zu löschen.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.model.DeleteLedgerRequest;
import com.amazonaws.services.qldb.model.DeleteLedgerResult;
import com.amazonaws.services.qldb.model.ResourceNotFoundException;
import com.amazonaws.services.qldb.model.UpdateLedgerRequest;
import com.amazonaws.services.qldb.model.UpdateLedgerResult;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
```

```
 * Delete a ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class DeleteLedger {
    public static final Logger log = LoggerFactory.getLogger(DeleteLedger.class);
    public static final Long LEDGER_DELETION_POLL_PERIOD_MS = 20_000L;
    public static AmazonQLDB client = CreateLedger.getClient();

    private DeleteLedger() { }

    public static void main(String... args) throws Exception {
        try {
            setDeletionProtection(Constants.LEDGER_NAME, false);

            delete(Constants.LEDGER_NAME);

            waitForDeleted(Constants.LEDGER_NAME);

        } catch (Exception e) {
            log.error("Unable to delete the ledger.", e);
            throw e;
        }
    }

    /**
     * Send a request to the QLDB database to delete the specified ledger.
     *
     * @param ledgerName
     *              Name of the ledger to be deleted.
     * @return DeleteLedgerResult.
     */
    public static DeleteLedgerResult delete(final String ledgerName) {
        log.info("Attempting to delete the ledger with name: {}...", ledgerName);
        DeleteLedgerRequest request = new
 DeleteLedgerRequest().withName(ledgerName);
        DeleteLedgerResult result = client.deleteLedger(request);
        log.info("Success.");
        return result;
    }

    /**
```

```
     * Wait for the ledger to be deleted.
     *
     * @param ledgerName
     *              Name of the ledger being deleted.
     * @throws InterruptedException if thread is being interrupted.
     */
    public static void waitForDeleted(final String ledgerName) throws
 InterruptedException {
        log.info("Waiting for the ledger to be deleted...");
        while (true) {
            try {
                DescribeLedger.describe(ledgerName);
                log.info("The ledger is still being deleted. Please wait...");
                Thread.sleep(LEDGER_DELETION_POLL_PERIOD_MS);
            } catch (ResourceNotFoundException ex) {
                log.info("Success. The ledger is deleted.");
                break;
            }
        }
    }

    public static UpdateLedgerResult setDeletionProtection(String ledgerName,
 boolean deletionProtection) {
        log.info("Let's set deletionProtection to {} for the ledger with name {}",
 deletionProtection, ledgerName);
        UpdateLedgerRequest request = new UpdateLedgerRequest()
                .withName(ledgerName)
                .withDeletionProtection(deletionProtection);

        UpdateLedgerResult result = client.updateLedger(request);
        log.info("Success. Ledger updated: {}", result);
        return result;
    }
}
```

> **Note**
>
> Wenn der Löschschutz für Ihr Ledger aktiviert ist, müssen Sie ihn zuerst deaktivieren, bevor Sie das Ledger mithilfe der QLDB-API löschen können.

2. Wenn Sie im vorherigen Schritt Journaldaten exportiert haben und diese nicht mehr benötigen, verwenden Sie die Amazon S3 S3-Konsole, um Ihren S3-Bucket zu löschen.

Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.

Anleitung zu Amazon QLDB Node.js

In dieser Implementierung der Tutorial-Beispielanwendung verwenden Sie den Amazon QLDB-Treiber mit dem AWS SDK für JavaScript in Node.js, um ein QLDB-Ledger zu erstellen und es mit Beispieldaten zu füllen.

Während Sie dieses Tutorial durcharbeiten, können Sie sich auf die API-Referenz für AWS SDK for JavaScriptVerwaltungs-API-Operationen beziehen. Informationen zu Transaktionsdatenoperationen finden Sie in der API-Referenz zum QLDB-Treiber für Node.js.

> **Note**
>
> Gegebenenfalls enthalten einige Tutorialschritte unterschiedliche Befehle oder Codebeispiele für jede unterstützte Hauptversion des QLDB-Treibers für Node.js.

Themen

Installation der Amazon QLDB-Beispielanwendung Node.js

In diesem Abschnitt wird beschrieben, wie Sie die bereitgestellte Amazon QLDB-Beispielanwendung für das Tutorial step-by-step Node.js installieren und ausführen. Der Anwendungsfall für diese Beispielanwendung ist eine Datenbank der Straßenverkehrsbehörde, mit der die vollständigen Verlaufsinformationen über Fahrzeugzulassungen nachverfolgt werden.

Die DMV-Beispielanwendung für Node.js ist Open Source im GitHub Repository amazon-qldb-dmv-sampleaws-samples/ -nodejs.

Voraussetzungen

Bevor Sie beginnen, stellen Sie sicher, dass Sie den QLDB-Treiber für Node.js fertiggestellt haben. Voraussetzungen Dazu gehört die Installation von Node.js und die folgenden Schritte:

1. Melde dich an fürAWS.
2. Erstellen Sie einen Benutzer mit den entsprechenden QLDB-Berechtigungen.
3. Gewähren Sie programmatischen Zugriff für die Entwicklung.

Um alle Schritte in diesem Tutorial ausführen zu können, benötigen Sie vollen Administratorzugriff auf Ihre Ledger-Ressource über die QLDB-API.

Installation

So installieren Sie die Beispielanwendung

1. Geben Sie den folgenden Befehl ein, um die Beispielanwendung von zu klonen. GitHub

   2.x

   ```
   git clone https://github.com/aws-samples/amazon-qldb-dmv-sample-nodejs.git
   ```

   1.x

   ```
   git clone -b v1.0.0 https://github.com/aws-samples/amazon-qldb-dmv-sample-
   nodejs.git
   ```

   Die Beispielanwendung packt den vollständigen Quellcode aus diesem Tutorial und seine Abhängigkeiten, einschließlich des Treibers Node.js und des AWSSDK für JavaScript in Node.js. Diese Anwendung ist in geschrieben TypeScript.
2. Wechseln Sie zu dem Verzeichnis, in dem das `amazon-qldb-dmv-sample-nodejs`-Paket geklont wird.

   ```
   cd amazon-qldb-dmv-sample-nodejs
   ```

3. Führen Sie eine Neueinstallation der Abhängigkeiten durch.

```
npm ci
```

4. Transpilieren Sie das Paket.

```
npm run build
```

   Die transpilierten JavaScript Dateien werden in das `./dist` Verzeichnis geschrieben.

5. Fahren Sie mit Schritt 1: Erstellen Sie ein neues Hauptbuch fort, um das Tutorial zu starten und ein Ledger zu erstellen.

Schritt 1: Erstellen Sie ein neues Hauptbuch

In diesem Schritt erstellen Sie ein neues Amazon QLDB-Ledger mit dem Namen. `vehicle-registration`

So erstellen Sie einen neuen Ledger

1. Überprüfen Sie die folgende Datei (`Constants.ts`), die konstante Werte enthält, die von allen anderen Programmen in diesem Tutorial verwendet werden.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

/**
 * Constant values used throughout this tutorial.
 */
export const LEDGER_NAME = "vehicle-registration";
export const LEDGER_NAME_WITH_TAGS = "tags";

export const DRIVERS_LICENSE_TABLE_NAME = "DriversLicense";
export const PERSON_TABLE_NAME = "Person";
export const VEHICLE_REGISTRATION_TABLE_NAME = "VehicleRegistration";
export const VEHICLE_TABLE_NAME = "Vehicle";

export const GOV_ID_INDEX_NAME = "GovId";
export const LICENSE_NUMBER_INDEX_NAME = "LicenseNumber";
export const LICENSE_PLATE_NUMBER_INDEX_NAME = "LicensePlateNumber";
export const PERSON_ID_INDEX_NAME = "PersonId";
export const VIN_INDEX_NAME = "VIN";

export const RETRY_LIMIT = 4;

export const JOURNAL_EXPORT_S3_BUCKET_NAME_PREFIX = "qldb-tutorial-journal-export";
export const USER_TABLES = "information_schema.user_tables";
```

2. Verwenden Sie das folgende Programm (`CreateLedger.ts`), um einen Ledger mit dem Namen `vehicle-registration` zu erstellen.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
```

```
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QLDB } from "aws-sdk";
import {
    CreateLedgerRequest,
    CreateLedgerResponse,
    DescribeLedgerRequest,
    DescribeLedgerResponse
} from "aws-sdk/clients/qldb";

import { LEDGER_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { sleep } from "./qldb/Util";

const LEDGER_CREATION_POLL_PERIOD_MS = 10000;
const ACTIVE_STATE = "ACTIVE";

/**
 * Create a new ledger with the specified name.
 * @param ledgerName Name of the ledger to be created.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with a CreateLedgerResponse.
 */
export async function createLedger(ledgerName: string, qldbClient: QLDB):
 Promise<CreateLedgerResponse> {
    log(`Creating a ledger named: ${ledgerName}...`);
    const request: CreateLedgerRequest = {
        Name: ledgerName,
        PermissionsMode: "ALLOW_ALL"
    }
    const result: CreateLedgerResponse = await
 qldbClient.createLedger(request).promise();
    log(`Success. Ledger state: ${result.State}.`);
```

```
    return result;
}

/**
 * Wait for the newly created ledger to become active.
 * @param ledgerName Name of the ledger to be checked on.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with a DescribeLedgerResponse.
 */
export async function waitForActive(ledgerName: string, qldbClient: QLDB):
 Promise<DescribeLedgerResponse> {
    log(`Waiting for ledger ${ledgerName} to become active...`);
    const request: DescribeLedgerRequest = {
        Name: ledgerName
    }
    while (true) {
        const result: DescribeLedgerResponse = await
 qldbClient.describeLedger(request).promise();
        if (result.State === ACTIVE_STATE) {
            log("Success. Ledger is active and ready to be used.");
            return result;
        }
        log("The ledger is still creating. Please wait...");
        await sleep(LEDGER_CREATION_POLL_PERIOD_MS);
    }
}

/**
 * Create a ledger and wait for it to be active.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbClient: QLDB = new QLDB();
        await createLedger(LEDGER_NAME, qldbClient);
        await waitForActive(LEDGER_NAME, qldbClient);
    } catch (e) {
        error(`Unable to create the ledger: ${e}`);
    }
}

if (require.main === module) {
    main();
```

```
}
```

> **Note**
>
> - Im `createLedger`-Aufruf müssen Sie einen Ledger-Namen und einen Berechtigungsmodus angeben. Wir empfehlen, den `STANDARD` Berechtigungsmodus zu verwenden, um die Sicherheit Ihrer Ledger-Daten zu maximieren.
> - Wenn Sie einen Ledger erstellen, ist der Löschschutz standardmäßig aktiviert. Dies ist eine Funktion in QLDB, die verhindert, dass Ledger von Benutzern gelöscht werden. Sie haben die Möglichkeit, den Löschschutz bei der Ledger-Erstellung mithilfe der QLDB-API oder der () zu deaktivieren. AWS Command Line Interface AWS CLI
> - Optional können Sie auch Tags angeben, die Ihrem Ledger angefügt werden sollen.

3. Geben Sie den folgenden Befehl ein, um das transpilierte Programm auszuführen.

```
node dist/CreateLedger.js
```

Fahren Sie zum Überprüfen der Verbindung mit dem neuen Ledger mit Schritt 2: Testen Sie die Konnektivität zum Ledger fort.

Schritt 2: Testen Sie die Konnektivität zum Ledger

In diesem Schritt überprüfen Sie, ob Sie über den Transaktionsdaten-API-Endpunkt eine Verbindung zum `vehicle-registration` Ledger in Amazon QLDB herstellen können.

So testen Sie die Verbindung zum Ledger

1. Verwenden Sie das folgende Programm (`ConnectToLedger.ts`), um eine Datensitzungsverbindung mit dem Ledger `vehicle-registration` zu erstellen.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
```

```
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, RetryConfig  } from "amazon-qldb-driver-nodejs";
import { ClientConfiguration } from "aws-sdk/clients/qldbsession";

import { LEDGER_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

const qldbDriver: QldbDriver = createQldbDriver();

/**
 * Create a driver for creating sessions.
 * @param ledgerName The name of the ledger to create the driver on.
 * @param serviceConfigurationOptions The configurations for the AWS SDK client
 that the driver uses.
 * @returns The driver for creating sessions.
 */
export function createQldbDriver(
    ledgerName: string = LEDGER_NAME,
    serviceConfigurationOptions: ClientConfiguration = {}
): QldbDriver {
    const retryLimit = 4;
    const maxConcurrentTransactions = 10;
    //Use driver's default backoff function (and hence, no second parameter
 provided to RetryConfig)
    const retryConfig: RetryConfig = new RetryConfig(retryLimit);
```

```
    const qldbDriver: QldbDriver = new QldbDriver(ledgerName,
 serviceConfigurationOptions, maxConcurrentTransactions, retryConfig);
    return qldbDriver;
}

export function getQldbDriver(): QldbDriver {
    return qldbDriver;
}

/**
 * Connect to a session for a given ledger using default settings.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        log("Listing table names...");
        const tableNames: string[] = await qldbDriver.getTableNames();
        tableNames.forEach((tableName: string): void => {
            log(tableName);
        });
    } catch (e) {
        error(`Unable to create session: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

1.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
```

```
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver  } from "amazon-qldb-driver-nodejs";
import { ClientConfiguration } from "aws-sdk/clients/qldbsession";

import { LEDGER_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

const qldbDriver: QldbDriver = createQldbDriver();

/**
 * Create a driver for creating sessions.
 * @param ledgerName The name of the ledger to create the driver on.
 * @param serviceConfigurationOptions The configurations for the AWS SDK client
 that the driver uses.
 * @returns The driver for creating sessions.
 */
export function createQldbDriver(
    ledgerName: string = LEDGER_NAME,
    serviceConfigurationOptions: ClientConfiguration = {}
): QldbDriver {
    const qldbDriver: QldbDriver = new QldbDriver(ledgerName,
 serviceConfigurationOptions);
    return qldbDriver;
}

export function getQldbDriver(): QldbDriver {
    return qldbDriver;
}

/**
 * Connect to a session for a given ledger using default settings.
```

```
 * @returns Promise which fulfills with void.
 */
var main = async function(): Promise<void> {
    try {
        log("Listing table names...");
        const tableNames: string[] = await qldbDriver.getTableNames();
        tableNames.forEach((tableName: string): void => {
            log(tableName);
        });
    } catch (e) {
        error(`Unable to create session: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Um Datentransaktionen in Ihrem Ledger auszuführen, müssen Sie ein QLDB-Treiberobjekt erstellen, um eine Verbindung zu einem bestimmten Ledger herzustellen. Dies ist ein anderes Client-Objekt als das `qldbClient`-Objekt, das Sie im vorherigen Schritt zum Erstellen des Ledgers verwendet haben. Dieser vorherige Client wird nur verwendet, um die in der aufgeführten Verwaltungs-API-Operationen auszuführen. Amazon QLDB API-Referenz

2. Geben Sie den folgenden Befehl ein, um das transpilierte Programm auszuführen.

```
node dist/ConnectToLedger.js
```

Fahren Sie zum Erstellen von Tabellen im `vehicle-registration`-Ledger mit Schritt 3: Erstellen Sie Tabellen, Indizes und Beispieldaten fort.

Schritt 3: Erstellen Sie Tabellen, Indizes und Beispieldaten

Wenn Ihr Amazon QLDB-Ledger aktiv ist und Verbindungen akzeptiert, können Sie damit beginnen, Tabellen mit Daten über Fahrzeuge, deren Besitzer und deren Zulassungsinformationen zu erstellen. Nach dem Erstellen der Tabellen und Indizes können Sie Daten in diese laden.

In diesem Schritt erstellen Sie vier Tabellen im `vehicle-registration`-Ledger:

- `VehicleRegistration`
- `Vehicle`
- `Person`
- `DriversLicense`

Außerdem erzeugen Sie die folgenden Indizes.

Tabellenname	Feld
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Beim Einfügen von Beispieldaten fügen Sie zunächst Dokumente in die `Person`-Tabelle ein. Anschließend verwenden Sie die vom System zugewiesene `id` aus den `Person`-Dokumenten, um die entsprechenden Felder in den relevanten `VehicleRegistration`- und `DriversLicense`-Dokumenten auszufüllen.

> **Tip**
>
> Es hat sich bewährt, das vom System zugewiesene `id` Dokument als Fremdschlüssel zu verwenden. Sie können zwar Felder definieren, die eindeutige Kennungen (z. B. eine Kfz-VIN) sein sollen, die echte eindeutige Kennung eines Dokuments ist aber seine `id`. Dieses Feld ist in den Metadaten des Dokuments enthalten, die Sie in der festgeschriebenen Ansicht (d. h. in der vom System definierten Ansicht der Tabelle) abrufen können.
> Weitere Hinweise zu Ansichten in QLDB finden Sie unter. Schlüsselkonzepte Weitere Informationen zu Metadaten finden Sie unter Metadaten von Dokumenten abfragen.

So erstellen Sie Tabellen und Indizes

1. Verwenden Sie das folgende Programm (`CreateTable.ts`), um die oben genannten Tabellen zu erstellen.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";

import { getQldbDriver } from "./ConnectToLedger";
import {
    DRIVERS_LICENSE_TABLE_NAME,
    PERSON_TABLE_NAME,
    VEHICLE_REGISTRATION_TABLE_NAME,
    VEHICLE_TABLE_NAME
} from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

/**
```

```
 * Create multiple tables in a single transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName Name of the table to create.
 * @returns Promise which fulfills with the number of changes to the database.
 */
export async function createTable(txn: TransactionExecutor, tableName: string):
 Promise<number> {
    const statement: string = `CREATE TABLE ${tableName}`;
    return await txn.execute(statement).then((result: Result) => {
        log(`Successfully created table ${tableName}.`);
        return result.getResultList().length;
    });
}

/**
 * Create tables in a QLDB ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            Promise.all([
                createTable(txn, VEHICLE_REGISTRATION_TABLE_NAME),
                createTable(txn, VEHICLE_TABLE_NAME),
                createTable(txn, PERSON_TABLE_NAME),
                createTable(txn, DRIVERS_LICENSE_TABLE_NAME)
            ]);
        });
    } catch (e) {
        error(`Unable to create tables: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Dieses Programm demonstriert, wie die `executeLambda` Funktion in einer QLDB-Treiberinstanz verwendet wird. In diesem Beispiel führen Sie mehrere `CREATE TABLE`-PartiQL-Anweisungen mit einem einzelnen Lambda-Ausdruck aus.

Diese Ausführungsfunktion startet implizit eine Transaktion, führt alle Anweisungen im Lambda aus und schreibt dann die Transaktion automatisch fest.

2. Geben Sie den folgenden Befehl ein, um das transpilierte Programm auszuführen.

```
node dist/CreateTable.js
```

3. Verwenden Sie das folgende Programm (`CreateIndex.ts`), um wie zuvor beschrieben Indizes für die Tabellen zu erstellen.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";

import { getQldbDriver } from "./ConnectToLedger";
import {
    DRIVERS_LICENSE_TABLE_NAME,
    GOV_ID_INDEX_NAME,
    LICENSE_NUMBER_INDEX_NAME,
    LICENSE_PLATE_NUMBER_INDEX_NAME,
```

```
    PERSON_ID_INDEX_NAME,
    PERSON_TABLE_NAME,
    VEHICLE_REGISTRATION_TABLE_NAME,
    VEHICLE_TABLE_NAME,
    VIN_INDEX_NAME
} from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

/**
 * Create an index for a particular table.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName Name of the table to add indexes for.
 * @param indexAttribute Index to create on a single attribute.
 * @returns Promise which fulfills with the number of changes to the database.
 */
export async function createIndex(
    txn: TransactionExecutor,
    tableName: string,
    indexAttribute: string
): Promise<number> {
    const statement: string = `CREATE INDEX on ${tableName} (${indexAttribute})`;
    return await txn.execute(statement).then((result) => {
        log(`Successfully created index ${indexAttribute} on table ${tableName}.`);
        return result.getResultList().length;
    });
}

/**
 * Create indexes on tables in a particular ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            Promise.all([
                createIndex(txn, PERSON_TABLE_NAME, GOV_ID_INDEX_NAME),
                createIndex(txn, VEHICLE_TABLE_NAME, VIN_INDEX_NAME),
                createIndex(txn, VEHICLE_REGISTRATION_TABLE_NAME, VIN_INDEX_NAME),
                createIndex(txn, VEHICLE_REGISTRATION_TABLE_NAME,
 LICENSE_PLATE_NUMBER_INDEX_NAME),
                createIndex(txn, DRIVERS_LICENSE_TABLE_NAME, PERSON_ID_INDEX_NAME),
                createIndex(txn, DRIVERS_LICENSE_TABLE_NAME,
 LICENSE_NUMBER_INDEX_NAME)
```

```
            ]);
        });
    } catch (e) {
        error(`Unable to create indexes: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

4. Geben Sie den folgenden Befehl ein, um das transpilierte Programm auszuführen.

```
node dist/CreateIndex.js
```

So laden Sie Daten in die Tabellen

1. Überprüfen Sie die folgenden `.ts`-Dateien:

   1. `SampleData.ts`— Enthält die Beispieldaten, die Sie in die Tabellen einfügen. `vehicle-registration`

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { Decimal } from "ion-js";

const EMPTY_SECONDARY_OWNERS: object[] = [];
export const DRIVERS_LICENSE = [
    {
        PersonId: "",
        LicenseNumber: "LEWISR261LL",
        LicenseType: "Learner",
        ValidFromDate: new Date("2016-12-20"),
        ValidToDate: new Date("2020-11-15")
    },
    {
        PersonId: "",
        LicenseNumber : "LOGANB486CG",
        LicenseType: "Probationary",
        ValidFromDate : new Date("2016-04-06"),
        ValidToDate : new Date("2020-11-15")
    },
    {
        PersonId: "",
        LicenseNumber : "744 849 301",
        LicenseType: "Full",
        ValidFromDate : new Date("2017-12-06"),
        ValidToDate : new Date("2022-10-15")
    },
    {
        PersonId: "",
        LicenseNumber : "P626-168-229-765",
        LicenseType: "Learner",
        ValidFromDate : new Date("2017-08-16"),
        ValidToDate : new Date("2021-11-15")
    },
    {
        PersonId: "",
        LicenseNumber : "S152-780-97-415-0",
        LicenseType: "Probationary",
        ValidFromDate : new Date("2015-08-15"),
        ValidToDate : new Date("2021-08-21")
```

```
    }
];
export const PERSON = [
    {
        FirstName : "Raul",
        LastName : "Lewis",
        DOB : new Date("1963-08-19"),
        Address : "1719 University Street, Seattle, WA, 98109",
        GovId : "LEWISR261LL",
        GovIdType : "Driver License"
    },
    {
        FirstName : "Brent",
        LastName : "Logan",
        DOB : new Date("1967-07-03"),
        Address : "43 Stockert Hollow Road, Everett, WA, 98203",
        GovId : "LOGANB486CG",
        GovIdType : "Driver License"
    },
    {
        FirstName : "Alexis",
        LastName : "Pena",
        DOB : new Date("1974-02-10"),
        Address : "4058 Melrose Street, Spokane Valley, WA, 99206",
        GovId : "744 849 301",
        GovIdType : "SSN"
    },
    {
        FirstName : "Melvin",
        LastName : "Parker",
        DOB : new Date("1976-05-22"),
        Address : "4362 Ryder Avenue, Seattle, WA, 98101",
        GovId : "P626-168-229-765",
        GovIdType : "Passport"
    },
    {
        FirstName : "Salvatore",
        LastName : "Spencer",
        DOB : new Date("1997-11-15"),
        Address : "4450 Honeysuckle Lane, Seattle, WA, 98101",
        GovId : "S152-780-97-415-0",
        GovIdType : "Passport"
    }
];
```

```
export const VEHICLE = [
    {
        VIN : "1N4AL11D75C109151",
        Type : "Sedan",
        Year : 2011,
        Make : "Audi",
        Model : "A5",
        Color : "Silver"
    },
    {
        VIN : "KM8SRDHF6EU074761",
        Type : "Sedan",
        Year : 2015,
        Make : "Tesla",
        Model : "Model S",
        Color : "Blue"
    },
    {
        VIN : "3HGGK5G53FM761765",
        Type : "Motorcycle",
        Year : 2011,
        Make : "Ducati",
        Model : "Monster 1200",
        Color : "Yellow"
    },
    {
        VIN : "1HVBBAANXWH544237",
        Type : "Semi",
        Year : 2009,
        Make : "Ford",
        Model : "F 150",
        Color : "Black"
    },
    {
        VIN : "1C4RJFAG0FC625797",
        Type : "Sedan",
        Year : 2019,
        Make : "Mercedes",
        Model : "CLK 350",
        Color : "White"
    }
];
export const VEHICLE_REGISTRATION = [
    {
```

```
        VIN : "1N4AL11D75C109151",
        LicensePlateNumber : "LEWISR261LL",
        State : "WA",
        City : "Seattle",
        ValidFromDate : new Date("2017-08-21"),
        ValidToDate : new Date("2020-05-11"),
        PendingPenaltyTicketAmount : new Decimal(9025, -2),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    },
    {
        VIN : "KM8SRDHF6EU074761",
        LicensePlateNumber : "CA762X",
        State : "WA",
        City : "Kent",
        PendingPenaltyTicketAmount : new Decimal(13075, -2),
        ValidFromDate : new Date("2017-09-14"),
        ValidToDate : new Date("2020-06-25"),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    },
    {
        VIN : "3HGGK5G53FM761765",
        LicensePlateNumber : "CD820Z",
        State : "WA",
        City : "Everett",
        PendingPenaltyTicketAmount : new Decimal(44230, -2),
        ValidFromDate : new Date("2011-03-17"),
        ValidToDate : new Date("2021-03-24"),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    },
    {
        VIN : "1HVBBAANXWH544237",
        LicensePlateNumber : "LS477D",
        State : "WA",
        City : "Tacoma",
        PendingPenaltyTicketAmount : new Decimal(4220, -2),
```

```
        ValidFromDate : new Date("2011-10-26"),
        ValidToDate : new Date("2023-09-25"),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    },
    {
        VIN : "1C4RJFAG0FC625797",
        LicensePlateNumber : "TH393F",
        State : "WA",
        City : "Olympia",
        PendingPenaltyTicketAmount : new Decimal(3045, -2),
        ValidFromDate : new Date("2013-09-02"),
        ValidToDate : new Date("2024-03-19"),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    }
];
```

2. `Util.ts`— Ein Hilfsmodul, das aus dem `ion-js` Paket importiert, um Hilfsfunktionen bereitzustellen, die Amazon Ion-Daten konvertieren, analysieren und drucken.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
```

```
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { Result, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { GetBlockResponse, GetDigestResponse, ValueHolder } from "aws-sdk/
clients/qldb";
import {
    Decimal,
    decodeUtf8,
    dom,
    IonTypes,
    makePrettyWriter,
    makeReader,
    Reader,
    Timestamp,
    toBase64,
    Writer
} from "ion-js";

import { error } from "./LogUtil";



/**
 * TODO: Replace this with json.stringify
 * Returns the string representation of a given BlockResponse.
 * @param blockResponse The BlockResponse to convert to string.
 * @returns The string representation of the supplied BlockResponse.
 */
export function blockResponseToString(blockResponse: GetBlockResponse): string {
    let stringBuilder: string = "";
    if (blockResponse.Block.IonText) {
        stringBuilder = stringBuilder + "Block: " + blockResponse.Block.IonText +
 ", ";
    }
    if (blockResponse.Proof.IonText) {
        stringBuilder = stringBuilder + "Proof: " + blockResponse.Proof.IonText;
    }
    stringBuilder = "{" + stringBuilder + "}";
```

```
    const writer: Writer = makePrettyWriter();
    const reader: Reader = makeReader(stringBuilder);
    writer.writeValues(reader);
    return decodeUtf8(writer.getBytes());
}

/**
 * TODO: Replace this with json.stringify
 * Returns the string representation of a given GetDigestResponse.
 * @param digestResponse The GetDigestResponse to convert to string.
 * @returns The string representation of the supplied GetDigestResponse.
 */
export function digestResponseToString(digestResponse: GetDigestResponse): string
 {
    let stringBuilder: string = "";
    if (digestResponse.Digest) {
        stringBuilder += "Digest: " + JSON.stringify(toBase64(<Uint8Array>
 digestResponse.Digest)) + ", ";
    }
    if (digestResponse.DigestTipAddress.IonText) {
        stringBuilder += "DigestTipAddress: " +
 digestResponse.DigestTipAddress.IonText;
    }
    stringBuilder = "{" + stringBuilder + "}";
    const writer: Writer = makePrettyWriter();
    const reader: Reader = makeReader(stringBuilder);
    writer.writeValues(reader);
    return decodeUtf8(writer.getBytes());
}

/**
 * Get the document IDs from the given table.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName The table name to query.
 * @param field A field to query.
 * @param value The key of the given field.
 * @returns Promise which fulfills with the document ID as a string.
 */
export async function getDocumentId(
    txn: TransactionExecutor,
    tableName: string,
    field: string,
    value: string
): Promise<string> {
```

```
    const query: string = `SELECT id FROM ${tableName} AS t BY id WHERE t.
${field} = ?`;
    let documentId: string = undefined;
    await txn.execute(query, value).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        if (resultList.length === 0) {
            throw new Error(`Unable to retrieve document ID using ${value}.`);
        }
        documentId = resultList[0].get("id").stringValue();

    }).catch((err: any) => {
        error(`Error getting documentId: ${err}`);
    });
    return documentId;
}

/**
 * Sleep for the specified amount of time.
 * @param ms The amount of time to sleep in milliseconds.
 * @returns Promise which fulfills with void.
 */
export function sleep(ms: number): Promise<void> {
    return new Promise(resolve => setTimeout(resolve, ms));
}

/**
 * Find the value of a given path in an Ion value. The path should contain a blob
 value.
 * @param value The Ion value that contains the journal block attributes.
 * @param path The path to a certain attribute.
 * @returns Uint8Array value of the blob, or null if the attribute cannot be
 found in the Ion value
 *                  or is not of type Blob
 */
export function getBlobValue(value: dom.Value, path: string): Uint8Array | null {
    const attribute: dom.Value = value.get(path);
    if (attribute !== null && attribute.getType() === IonTypes.BLOB) {
        return attribute.uInt8ArrayValue();
    }
    return null;
}

/**
 * TODO: Replace this with json.stringify
```

```
 * Returns the string representation of a given ValueHolder.
 * @param valueHolder The ValueHolder to convert to string.
 * @returns The string representation of the supplied ValueHolder.
 */
export function valueHolderToString(valueHolder: ValueHolder): string {
    const stringBuilder: string = `{ IonText: ${valueHolder.IonText}}`;
    const writer: Writer = makePrettyWriter();
    const reader: Reader = makeReader(stringBuilder);
    writer.writeValues(reader);
    return decodeUtf8(writer.getBytes());
}

/**
 * Converts a given value to Ion using the provided writer.
 * @param value The value to convert to Ion.
 * @param ionWriter The Writer to pass the value into.
 * @throws Error: If the given value cannot be converted to Ion.
 */
export function writeValueAsIon(value: any, ionWriter: Writer): void {
    switch (typeof value) {
        case "string":
            ionWriter.writeString(value);
            break;
        case "boolean":
            ionWriter.writeBoolean(value);
            break;
        case "number":
                ionWriter.writeInt(value);
                break;
        case "object":
            if (Array.isArray(value)) {
                // Object is an array.
                ionWriter.stepIn(IonTypes.LIST);

                for (const element of value) {
                    writeValueAsIon(element, ionWriter);
                }

                ionWriter.stepOut();
            } else if (value instanceof Date) {
                // Object is a Date.
                ionWriter.writeTimestamp(Timestamp.parse(value.toISOString()));
            } else if (value instanceof Decimal) {
                // Object is a Decimal.
```

```
                ionWriter.writeDecimal(value);
            } else if (value === null) {
                ionWriter.writeNull(IonTypes.NULL);
            } else {
                // Object is a struct.
                ionWriter.stepIn(IonTypes.STRUCT);

                for (const key of Object.keys(value)) {
                    ionWriter.writeFieldName(key);
                    writeValueAsIon(value[key], ionWriter);
                }
                ionWriter.stepOut();
            }
            break;
        default:
            throw new Error(`Cannot convert to Ion for type: ${(typeof
 value)}.`);
    }
}
```

> **Note**
>
> Die `getDocumentId`-Funktion führt eine Abfrage aus, die vom System zugewiesene Dokument-IDs aus einer Tabelle zurückgibt. Weitere Informationen hierzu finden Sie unter Verwenden der BY-Klausel zur Abfrage der Dokument-ID.

2. Verwenden Sie das folgende Programm (`InsertDocument.ts`), um Beispieldaten in Ihre Tabellen einzufügen.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
```

```
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";
import { dom } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { DRIVERS_LICENSE, PERSON, VEHICLE, VEHICLE_REGISTRATION } from "./model/
SampleData";
import {
    DRIVERS_LICENSE_TABLE_NAME,
    PERSON_TABLE_NAME,
    VEHICLE_REGISTRATION_TABLE_NAME,
    VEHICLE_TABLE_NAME
} from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

/**
 * Insert the given list of documents into a table in a single transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName Name of the table to insert documents into.
 * @param documents List of documents to insert.
 * @returns Promise which fulfills with a {@linkcode Result} object.
 */
export async function insertDocument(
    txn: TransactionExecutor,
    tableName: string,
    documents: object[]
): Promise<Result> {
    const statement: string = `INSERT INTO ${tableName} ?`;
    const result: Result = await txn.execute(statement, documents);
    return result;
}
```

```
/**
 * Handle the insertion of documents and updating PersonIds all in a single
 transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @returns Promise which fulfills with void.
 */
async function updateAndInsertDocuments(txn: TransactionExecutor): Promise<void> {
    log("Inserting multiple documents into the 'Person' table...");
    const documentIds: Result = await insertDocument(txn, PERSON_TABLE_NAME,
 PERSON);

    const listOfDocumentIds: dom.Value[] = documentIds.getResultList();
    log("Updating PersonIds for 'DriversLicense' and PrimaryOwner for
 'VehicleRegistration'...");
    updatePersonId(listOfDocumentIds);

    log("Inserting multiple documents into the remaining tables...");
    await Promise.all([
        insertDocument(txn, DRIVERS_LICENSE_TABLE_NAME, DRIVERS_LICENSE),
        insertDocument(txn, VEHICLE_REGISTRATION_TABLE_NAME, VEHICLE_REGISTRATION),
        insertDocument(txn, VEHICLE_TABLE_NAME, VEHICLE)
    ]);
}

/**
 * Update the PersonId value for DriversLicense records and the PrimaryOwner value
 for VehicleRegistration records.
 * @param documentIds List of document IDs.
 */
export function updatePersonId(documentIds: dom.Value[]): void {
    documentIds.forEach((value: dom.Value, i: number) => {
        const documentId: string = value.get("documentId").stringValue();
        DRIVERS_LICENSE[i].PersonId = documentId;
        VEHICLE_REGISTRATION[i].Owners.PrimaryOwner.PersonId = documentId;
    });
}

/**
 * Insert documents into a table in a QLDB ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
```

```
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await updateAndInsertDocuments(txn);
        });
    } catch (e) {
        error(`Unable to insert documents: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> - Dieses Programm veranschaulicht, wie die `execute`-Funktion mit parametrisierten Werten aufgerufen wird. Zusätzlich zu der PartiQL-Anweisung, die Sie ausführen möchten, können Sie Datenparameter übergeben. Verwenden Sie ein Fragezeichen (`?`) als Variablenplatzhalter in Ihrer Anweisungszeichenfolge.
> - Wenn eine `INSERT`-Anweisung erfolgreich ist, wird die `id` jedes eingefügten Dokuments zurückgegeben.

3. Geben Sie den folgenden Befehl ein, um das transpilierte Programm auszuführen.

```
node dist/InsertDocument.js
```

Als Nächstes können Sie mit `SELECT`-Anweisungen Daten aus den Tabellen im `vehicle-registration`-Ledger lesen. Fahren Sie mit Schritt 4: Fragen Sie die Tabellen in einem Hauptbuch ab fort.

Schritt 4: Fragen Sie die Tabellen in einem Hauptbuch ab

Nachdem Sie Tabellen in einem Amazon QLDB-Ledger erstellt und mit Daten geladen haben, können Sie Abfragen ausführen, um die gerade eingegebenen Fahrzeugregistrierungsdaten zu überprüfen. QLDB verwendet PartiQL als Abfragesprache und Amazon Ion als dokumentenorientiertes Datenmodell.

PartiQL ist eine SQL-kompatible Open-Source-Abfragesprache, die für die Verwendung mit Ion erweitert wurde. Mit PartiSQL können Sie Ihre Daten mit vertrauten SQL-Operatoren einfügen,

abfragen und verwalten. Amazon Ion ist eine Obermenge von JSON. Ion ist ein dokumentenbasiertes Open-Source-Datenformat, das Ihnen die Flexibilität bietet, strukturierte, halbstrukturierte und verschachtelte Daten zu speichern und zu verarbeiten.

In diesem Schritt verwenden Sie `SELECT`-Anweisungen zum Lesen von Daten aus den Tabellen im `vehicle-registration`-Ledger.

> **Warning**
>
> Wenn Sie eine Abfrage in QLDB ohne indizierte Suche ausführen, wird ein vollständiger Tabellenscan aufgerufen. PartiQL unterstützt solche Abfragen, weil es SQL-kompatibel ist. Führen Sie jedoch keine Tabellenscans für Produktionsanwendungsfälle in QLDB aus. Tabellenscans können bei großen Tabellen zu Leistungsproblemen führen, einschließlich Parallelitätskonflikten und Transaktions-Timeouts.
> Um Tabellenscans zu vermeiden, müssen Sie Anweisungen mit einer `WHERE` Prädikatklausel ausführen, indem Sie einen Gleichheitsoperator für ein indiziertes Feld oder eine Dokument-ID verwenden, z. B. oder. `WHERE indexedField = 123` `WHERE indexedField IN (456, 789)` Weitere Informationen finden Sie unter Optimieren der Abfrageleistung.

So fragen Sie die Tabellen ab

1. Verwenden Sie das folgende Programm (`FindVehicles.ts`), um alle Fahrzeuge abzufragen, die für eine Person in Ihrem Ledger zugelassen sind.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
```

```
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";
import { dom } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { PERSON } from "./model/SampleData";
import { PERSON_TABLE_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { getDocumentId } from "./qldb/Util";
import { prettyPrintResultList } from "./ScanTable";

/**
 * Query 'Vehicle' and 'VehicleRegistration' tables using a unique document ID in
 one transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param govId The owner's government ID.
 * @returns Promise which fulfills with void.
 */
async function findVehiclesForOwner(txn: TransactionExecutor, govId: string):
 Promise<void> {
    const documentId: string = await getDocumentId(txn, PERSON_TABLE_NAME, "GovId",
 govId);
    const query: string = "SELECT Vehicle FROM Vehicle INNER JOIN
 VehicleRegistration AS r " +
                          "ON Vehicle.VIN = r.VIN WHERE
 r.Owners.PrimaryOwner.PersonId = ?";

    await txn.execute(query, documentId).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        log(`List of vehicles for owner with GovId: ${govId}`);
        prettyPrintResultList(resultList);
    });
}
```

```
/**
 * Find all vehicles registered under a person.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await findVehiclesForOwner(txn, PERSON[0].GovId);
        });
    } catch (e) {
        error(`Error getting vehicles for owner: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Zunächst fragt dieses Programm die `Person`-Tabelle nach dem Dokument mit `GovId` `LEWISR261LL` ab, um sein `id`-Metadatenfeld abzurufen.
> Anschließend verwendet es dieses Dokument `id` als Fremdschlüssel, um die `VehicleRegistration`-Tabelle nach `PrimaryOwner.PersonId` abzufragen. Es tritt führt außerdem `VehicleRegistration` mit der `Vehicle`-Tabelle im `VIN`-Feld zusammen.

2. Geben Sie den folgenden Befehl ein, um das transpilierte Programm auszuführen.

```
node dist/FindVehicles.js
```

Weitere Informationen zum Ändern von Dokumenten in den Tabellen im `vehicle-registration`-Ledger finden Sie unter Schritt 5: Dokumente in einem Ledger ändern.

Schritt 5: Dokumente in einem Ledger ändern

Da Sie nun über Daten verfügen, mit denen Sie arbeiten können, können Sie damit beginnen, Änderungen an Dokumenten im `vehicle-registration` Hauptbuch in Amazon QLDB

vorzunehmen. In diesem Schritt verdeutlichen die folgenden Codebeispiele, wie Sie Data Manipulation Language (DML)-Anweisungen ausführen. Diese Anweisungen aktualisieren den primären Eigentümer eines Fahrzeugs und fügen einem anderen Fahrzeug einen sekundären Eigentümer hinzu.

So ändern Sie Dokumente

1. Verwenden Sie das folgende Programm (`TransferVehicleOwnership.ts`), um den primären Besitzer des Fahrzeugs mit VIN `1N4AL11D75C109151` in Ihrem Ledger zu aktualisieren.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";
import { dom } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { PERSON, VEHICLE } from "./model/SampleData";
import { PERSON_TABLE_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
```

```
import { getDocumentId } from "./qldb/Util";

/**
 * Query a driver's information using the given ID.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param documentId The unique ID of a document in the Person table.
 * @returns Promise which fulfills with an Ion value containing the person.
 */
export async function findPersonFromDocumentId(txn: TransactionExecutor,
 documentId: string): Promise<dom.Value> {
    const query: string = "SELECT p.* FROM Person AS p BY pid WHERE pid = ?";

    let personId: dom.Value;
    await txn.execute(query, documentId).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        if (resultList.length === 0) {
            throw new Error(`Unable to find person with ID: ${documentId}.`);
        }
        personId = resultList[0];
    });
    return personId;
}

/**
 * Find the primary owner for the given VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin The VIN to find primary owner for.
 * @returns Promise which fulfills with an Ion value containing the primary owner.
 */
export async function findPrimaryOwnerForVehicle(txn: TransactionExecutor, vin:
 string): Promise<dom.Value> {
    log(`Finding primary owner for vehicle with VIN: ${vin}`);
    const query: string = "SELECT Owners.PrimaryOwner.PersonId FROM
 VehicleRegistration AS v WHERE v.VIN = ?";

    let documentId: string = undefined;
    await txn.execute(query, vin).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        if (resultList.length === 0) {
            throw new Error(`Unable to retrieve document ID using ${vin}.`);
        }
        const PersonIdValue: dom.Value = resultList[0].get("PersonId");
        if (PersonIdValue === null) {
            throw new Error(`Expected field name PersonId not found.`);
```

```
        }
        documentId = PersonIdValue.stringValue();
    });
    return findPersonFromDocumentId(txn, documentId);
}

/**
 * Update the primary owner for a vehicle using the given VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin The VIN for the vehicle to operate on.
 * @param documentId New PersonId for the primary owner.
 * @returns Promise which fulfills with void.
 */
async function updateVehicleRegistration(txn: TransactionExecutor, vin: string,
 documentId: string): Promise<void> {
    const statement: string = "UPDATE VehicleRegistration AS r SET
 r.Owners.PrimaryOwner.PersonId = ? WHERE r.VIN = ?";

    log(`Updating the primary owner for vehicle with VIN: ${vin}...`);
    await txn.execute(statement, documentId, vin).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        if (resultList.length === 0) {
            throw new Error("Unable to transfer vehicle, could not find
 registration.");
        }
        log(`Successfully transferred vehicle with VIN ${vin} to new owner.`);
    });
}

/**
 * Validate the current owner of the given vehicle and transfer its ownership to a
 new owner in a single transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin The VIN of the vehicle to transfer ownership of.
 * @param currentOwner The GovId of the current owner of the vehicle.
 * @param newOwner The GovId of the new owner of the vehicle.
 */
export async function validateAndUpdateRegistration(
    txn: TransactionExecutor,
    vin: string,
    currentOwner: string,
    newOwner: string
): Promise<void> {
    const primaryOwner: dom.Value = await findPrimaryOwnerForVehicle(txn, vin);
```

```
    const govIdValue: dom.Value = primaryOwner.get("GovId");
    if (govIdValue !== null && govIdValue.stringValue() !== currentOwner) {
        log("Incorrect primary owner identified for vehicle, unable to transfer.");
    }
    else {
        const documentId: string = await getDocumentId(txn, PERSON_TABLE_NAME,
 "GovId", newOwner);
        await updateVehicleRegistration(txn, vin, documentId);
        log("Successfully transferred vehicle ownership!");
    }
}

/**
 * Find primary owner for a particular vehicle's VIN.
 * Transfer to another primary owner for a particular vehicle's VIN.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();

        const vin: string = VEHICLE[0].VIN;
        const previousOwnerGovId: string = PERSON[0].GovId;
        const newPrimaryOwnerGovId: string = PERSON[1].GovId;

        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await validateAndUpdateRegistration(txn, vin, previousOwnerGovId,
 newPrimaryOwnerGovId);
        });
    } catch (e) {
        error(`Unable to connect and run queries: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

2. Geben Sie den folgenden Befehl ein, um das transpilierte Programm auszuführen.

```
node dist/TransferVehicleOwnership.js
```

3. Verwenden Sie das folgende Programm (`AddSecondaryOwner.ts`), um dem Fahrzeug mit VIN `KM8SRDHF6EU074761` in Ihrem Ledger einen sekundären Besitzer hinzuzufügen.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";
import { dom } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { PERSON, VEHICLE_REGISTRATION } from "./model/SampleData";
import { PERSON_TABLE_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { getDocumentId } from "./qldb/Util";
import { prettyPrintResultList } from "./ScanTable";

/**
 * Add a secondary owner into 'VehicleRegistration' table for a particular VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin VIN of the vehicle to query.
```

```
 * @param secondaryOwnerId The secondary owner's person ID.
 * @returns Promise which fulfills with void.
 */
export async function addSecondaryOwner(
    txn: TransactionExecutor,
    vin: string,
    secondaryOwnerId: string
): Promise<void> {
    log(`Inserting secondary owner for vehicle with VIN: ${vin}`);
    const query: string =
        `FROM VehicleRegistration AS v WHERE v.VIN = ? INSERT INTO
 v.Owners.SecondaryOwners VALUE ?`;

    const personToInsert = {PersonId: secondaryOwnerId};
    await txn.execute(query, vin, personToInsert).then(async (result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        log("VehicleRegistration Document IDs which had secondary owners added: ");
        prettyPrintResultList(resultList);
    });
}

/**
 * Query for a document ID with a government ID.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param governmentId The government ID to query with.
 * @returns Promise which fulfills with the document ID as a string.
 */
export async function getDocumentIdByGovId(txn: TransactionExecutor, governmentId:
 string): Promise<string> {
    const documentId: string = await getDocumentId(txn, PERSON_TABLE_NAME, "GovId",
 governmentId);
    return documentId;
}

/**
 * Check whether a driver has already been registered for the given VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin VIN of the vehicle to query.
 * @param secondaryOwnerId The secondary owner's person ID.
 * @returns Promise which fulfills with a boolean.
 */
export async function isSecondaryOwnerForVehicle(
    txn: TransactionExecutor,
    vin: string,
```

```
    secondaryOwnerId: string
): Promise<boolean> {
    log(`Finding secondary owners for vehicle with VIN: ${vin}`);
    const query: string = "SELECT Owners.SecondaryOwners FROM VehicleRegistration
 AS v WHERE v.VIN = ?";

    let doesExist: boolean = false;

    await txn.execute(query, vin).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();

        resultList.forEach((value: dom.Value) => {
            const secondaryOwnersList: dom.Value[] =
 value.get("SecondaryOwners").elements();

            secondaryOwnersList.forEach((secondaryOwner) => {
                const personId: dom.Value = secondaryOwner.get("PersonId");
                if (personId !== null &&  personId.stringValue() ===
 secondaryOwnerId) {
                    doesExist = true;
                }
            });
        });
    });
    return doesExist;
}

/**
 * Finds and adds secondary owners for a vehicle.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        const vin: string = VEHICLE_REGISTRATION[1].VIN;
        const govId: string = PERSON[0].GovId;

        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            const documentId: string = await getDocumentIdByGovId(txn, govId);

            if (await isSecondaryOwnerForVehicle(txn, vin, documentId)) {
                log(`Person with ID ${documentId} has already been added as a
 secondary owner of this vehicle.`);
            } else {
```

```
                await addSecondaryOwner(txn, vin, documentId);
            }
        });

        log("Secondary owners successfully updated.");
    } catch (e) {
        error(`Unable to add secondary owner: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

4. Geben Sie den folgenden Befehl ein, um das transpilierte Programm auszuführen.

```
node dist/AddSecondaryOwner.js
```

Informationen zum Überprüfen dieser Änderungen im `vehicle-registration`-Ledger finden Sie unter Schritt 6: Den Revisionsverlauf für ein Dokument anzeigen.

Schritt 6: Den Revisionsverlauf für ein Dokument anzeigen

Nach dem Bearbeiten der Zulassungsdaten für ein Fahrzeug im vorherigen Schritt können Sie den Verlauf aller registrierten Eigentümer und alle anderen aktualisierten Felder abfragen. In diesem Schritt führen Sie eine Abfrage des Revisionsverlaufs eines Dokuments in der `VehicleRegistration`-Tabelle in Ihrem `vehicle-registration`-Ledger durch.

So zeigen Sie den Revisionsverlauf an

1. Verwenden Sie das folgende Programm (`QueryHistory.ts`), um den Versionsverlauf des `VehicleRegistration`-Dokuments mit VIN `1N4AL11D75C109151` abzufragen.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
```

```
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";
import { dom } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { VEHICLE_REGISTRATION } from "./model/SampleData";
import { VEHICLE_REGISTRATION_TABLE_NAME } from "./qldb/Constants";
import { prettyPrintResultList } from "./ScanTable";
import { error, log } from "./qldb/LogUtil";
import { getDocumentId } from "./qldb/Util";

/**
 * Find previous primary owners for the given VIN in a single transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin The VIN to find previous primary owners for.
 * @returns Promise which fulfills with void.
 */
async function previousPrimaryOwners(txn: TransactionExecutor, vin: string):
 Promise<void> {
    const documentId: string = await getDocumentId(txn,
 VEHICLE_REGISTRATION_TABLE_NAME, "VIN", vin);
    const todaysDate: Date = new Date();
    // set todaysDate back one minute to ensure end time is in the past
    // by the time the request reaches our backend
    todaysDate.setMinutes(todaysDate.getMinutes() - 1);
    const threeMonthsAgo: Date = new Date(todaysDate);
    threeMonthsAgo.setMonth(todaysDate.getMonth() - 3);
```

```
    const query: string =
        `SELECT data.Owners.PrimaryOwner, metadata.version FROM history ` +
        `(${VEHICLE_REGISTRATION_TABLE_NAME}, \`${threeMonthsAgo.toISOString()}\`,
 \`${todaysDate.toISOString()}\`) ` +
        `AS h WHERE h.metadata.id = ?`;

    await txn.execute(query, documentId).then((result: Result) => {
        log(`Querying the 'VehicleRegistration' table's history using VIN:
 ${vin}.`);
        const resultList: dom.Value[] = result.getResultList();
        prettyPrintResultList(resultList);
    });
}

/**
 * Query a table's history for a particular set of documents.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        const vin: string = VEHICLE_REGISTRATION[0].VIN;
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await previousPrimaryOwners(txn, vin);
        });
    } catch (e) {
        error(`Unable to query history to find previous owners: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

Note

- Sie können den Revisionsverlauf eines Dokuments anzeigen, indem Sie die integrierte Verlaufsfunktion in der folgenden Syntax abfragen.

  ```
  SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
  ```

```
[ WHERE h.metadata.id = 'id' ]
```

- Die Startzeit und die Endzeit sind beide optional. Es handelt sich um Amazon Ion-Literalwerte, die mit Backticks () bezeichnet werden können. `` `...` `` Weitere Informationen finden Sie unter Ion mit PartiQL in Amazon QLDB abfragen.
- Es hat sich bewährt, eine Verlaufsabfrage sowohl mit einem Datumsbereich (Start- und Endzeit) als auch mit einer Dokument-ID () zu qualifizieren. `metadata.id` QLDB verarbeitet `SELECT` Abfragen in Transaktionen, die einem Transaktions-Timeout-Limit unterliegen.

  Der QLDB-Verlauf wird nach Dokument-ID indexiert, und Sie können derzeit keine zusätzlichen Verlaufsindizes erstellen. Bei Verlaufsabfragen, die eine Start- und Endzeit enthalten, wird der Vorteil einer Qualifizierung nach einem bestimmten Zeitraum genutzt.

2. Geben Sie den folgenden Befehl ein, um das transpilierte Programm auszuführen.

```
node dist/QueryHistory.js
```

Fahren Sie zum kryptografischen Überprüfen einer Dokumentrevision im `vehicle-registration`-Ledger mit Schritt 7: Überprüfen Sie ein Dokument in einem Hauptbuch fort.

Schritt 7: Überprüfen Sie ein Dokument in einem Hauptbuch

Mit Amazon QLDB können Sie die Integrität eines Dokuments im Journal Ihres Hauptbuchs effizient überprüfen, indem Sie kryptografisches Hashing mit SHA-256 verwenden. Weitere Informationen darüber, wie Verifizierung und kryptografisches Hashing in QLDB funktionieren, finden Sie unter. Datenverifizierung in Amazon QLDB

In diesem Schritt überprüfen Sie eine Dokumentrevision in der Tabelle `VehicleRegistration` in Ihrem `vehicle-registration`-Ledger. Zuerst fordern Sie einen Digest an, der als Ausgabedatei zurückgegeben wird und als Signatur des gesamten Änderungsverlaufs Ihres Ledgers fungiert. Anschließend fordern Sie einen Nachweis für die Revision in Bezug auf diesen Digest an. Mit diesem Nachweis wird die Integrität Ihrer Revision verifiziert, wenn alle Validierungsprüfungen bestanden werden.

So überprüfen Sie eine Dokumentrevision

1. Überprüfen Sie die folgenden `.ts` Dateien, die die QLDB-Objekte enthalten, die für die Überprüfung erforderlich sind.

   1. `BlockAddress.ts`

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { ValueHolder } from "aws-sdk/clients/qldb";
import { dom, IonTypes } from "ion-js";

export class BlockAddress {
    _strandId: string;
    _sequenceNo: number;

    constructor(strandId: string, sequenceNo: number) {
        this._strandId = strandId;
        this._sequenceNo = sequenceNo;
    }
```

```
}

/**
 * Convert a block address from an Ion value into a ValueHolder.
 * Shape of the ValueHolder must be: {'IonText': "{strandId: <"strandId">,
 sequenceNo: <sequenceNo>}"}
 * @param value The Ion value that contains the block address values to convert.
 * @returns The ValueHolder that contains the strandId and sequenceNo.
 */
export function blockAddressToValueHolder(value: dom.Value): ValueHolder {
    const blockAddressValue : dom.Value = getBlockAddressValue(value);
    const strandId: string = getStrandId(blockAddressValue);
    const sequenceNo: number = getSequenceNo(blockAddressValue);
    const valueHolder: string = `{strandId: "${strandId}", sequenceNo:
 ${sequenceNo}}`;
    const blockAddress: ValueHolder = {IonText: valueHolder};
    return blockAddress;
}

/**
 * Helper method that to get the Metadata ID.
 * @param value The Ion value.
 * @returns The Metadata ID.
 */
export function getMetadataId(value: dom.Value): string {
    const metaDataId: dom.Value = value.get("id");
    if (metaDataId === null) {
        throw new Error(`Expected field name id, but not found.`);
    }
    return metaDataId.stringValue();
}

/**
 * Helper method to get the Sequence No.
 * @param value The Ion value.
 * @returns The Sequence No.
 */
export function getSequenceNo(value : dom.Value): number {
    const sequenceNo: dom.Value = value.get("sequenceNo");
    if (sequenceNo === null) {
        throw new Error(`Expected field name sequenceNo, but not found.`);
    }
    return sequenceNo.numberValue();
}
```

```
/**
 * Helper method to get the Strand ID.
 * @param value The Ion value.
 * @returns The Strand ID.
 */
export function getStrandId(value: dom.Value): string {
    const strandId: dom.Value = value.get("strandId");
    if (strandId === null) {
        throw new Error(`Expected field name strandId, but not found.`);
    }
    return strandId.stringValue();
}

export function getBlockAddressValue(value: dom.Value) : dom.Value {
    const type = value.getType();
    if (type !== IonTypes.STRUCT) {
        throw new Error(`Unexpected format: expected struct, but got IonType:
 ${type.name}`);
    }
    const blockAddress: dom.Value = value.get("blockAddress");
    if (blockAddress == null) {
        throw new Error(`Expected field name blockAddress, but not found.`);
    }
    return blockAddress;
}
```

2. Verifier.ts

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
```

```
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { Digest, ValueHolder } from "aws-sdk/clients/qldb";
import { createHash } from "crypto";
import { dom, toBase64 } from "ion-js";

import { getBlobValue } from "./Util";

const HASH_LENGTH: number = 32;
const UPPER_BOUND: number = 8;

/**
 * Build the candidate digest representing the entire ledger from the Proof
 hashes.
 * @param proof The Proof object.
 * @param leafHash The revision hash to pair with the first hash in the Proof
 hashes list.
 * @returns The calculated root hash.
 */
function buildCandidateDigest(proof: ValueHolder, leafHash: Uint8Array):
 Uint8Array {
    const parsedProof: Uint8Array[] = parseProof(proof);
    const rootHash: Uint8Array = calculateRootHashFromInternalHash(parsedProof,
 leafHash);
    return rootHash;
}

/**
 * Combine the internal hashes and the leaf hash until only one root hash
 remains.
 * @param internalHashes An array of hash values.
 * @param leafHash The revision hash to pair with the first hash in the Proof
 hashes list.
 * @returns The root hash constructed by combining internal hashes.
 */
```

```
function calculateRootHashFromInternalHash(internalHashes: Uint8Array[],
 leafHash: Uint8Array): Uint8Array {
    const rootHash: Uint8Array = internalHashes.reduce(joinHashesPairwise,
 leafHash);
    return rootHash;
}

/**
 * Compare two hash values by converting each Uint8Array byte, which is unsigned
 by default,
 * into a signed byte, assuming they are little endian.
 * @param hash1 The hash value to compare.
 * @param hash2 The hash value to compare.
 * @returns Zero if the hash values are equal, otherwise return the difference of
 the first pair of non-matching bytes.
 */
function compareHashValues(hash1: Uint8Array, hash2: Uint8Array): number {
    if (hash1.length !== HASH_LENGTH || hash2.length !== HASH_LENGTH) {
        throw new Error("Invalid hash.");
    }
    for (let i = hash1.length-1; i >= 0; i--) {
        const difference: number = (hash1[i]<<24 >>24) - (hash2[i]<<24 >>24);
        if (difference !== 0) {
            return difference;
        }
    }
    return 0;
}

/**
 * Helper method that concatenates two Uint8Array.
 * @param arrays List of array to concatenate, in the order provided.
 * @returns The concatenated array.
 */
function concatenate(...arrays: Uint8Array[]): Uint8Array {
    let totalLength = 0;
    for (const arr of arrays) {
        totalLength += arr.length;
    }
    const result = new Uint8Array(totalLength);
    let offset = 0;
    for (const arr of arrays) {
        result.set(arr, offset);
        offset += arr.length;
```

```
    }
    return result;
}

/**
 * Flip a single random bit in the given hash value.
 * This method is intended to be used for purpose of demonstrating the QLDB
 verification features only.
 * @param original The hash value to alter.
 * @returns The altered hash with a single random bit changed.
 */
export function flipRandomBit(original: any): Uint8Array {
    if (original.length === 0) {
        throw new Error("Array cannot be empty!");
    }
    const bytePos: number = Math.floor(Math.random() * original.length);
    const bitShift: number = Math.floor(Math.random() * UPPER_BOUND);
    const alteredHash: Uint8Array = original;

    alteredHash[bytePos] = alteredHash[bytePos] ^ (1 << bitShift);
    return alteredHash;
}

/**
 * Take two hash values, sort them, concatenate them, and generate a new hash
 value from the concatenated values.
 * @param h1 Byte array containing one of the hashes to compare.
 * @param h2 Byte array containing one of the hashes to compare.
 * @returns The concatenated array of hashes.
 */
export function joinHashesPairwise(h1: Uint8Array, h2: Uint8Array): Uint8Array {
    if (h1.length === 0) {
        return h2;
    }
    if (h2.length === 0) {
        return h1;
    }
    let concat: Uint8Array;
    if (compareHashValues(h1, h2) < 0) {
        concat = concatenate(h1, h2);
    } else {
        concat = concatenate(h2, h1);
    }
    const hash = createHash('sha256');
```

```
    hash.update(concat);
    const newDigest: Uint8Array = hash.digest();
    return newDigest;
}

/**
 * Parse the Block object returned by QLDB and retrieve block hash.
 * @param valueHolder A structure containing an Ion string value.
 * @returns The block hash.
 */
export function parseBlock(valueHolder: ValueHolder): Uint8Array {
    const block: dom.Value = dom.load(valueHolder.IonText);
    const blockHash: Uint8Array = getBlobValue(block, "blockHash");
    return blockHash;
}

/**
 * Parse the Proof object returned by QLDB into an iterator.
 * The Proof object returned by QLDB is a dictionary like the following:
 * {'IonText': '[{{<hash>}},{{<hash>}}]'}
 * @param valueHolder A structure containing an Ion string value.
 * @returns A list of hash values.
 */
function parseProof(valueHolder: ValueHolder): Uint8Array[] {
    const proofs : dom.Value = dom.load(valueHolder.IonText);
    return proofs.elements().map(proof => proof.uInt8ArrayValue());
}

/**
 *  Verify document revision against the provided digest.
 * @param documentHash The SHA-256 value representing the document revision to be
 verified.
 * @param digest The SHA-256 hash value representing the ledger digest.
 * @param proof The Proof object retrieved from GetRevision.getRevision.
 * @returns If the document revision verifies against the ledger digest.
 */
export function verifyDocument(documentHash: Uint8Array, digest: Digest, proof:
 ValueHolder): boolean {
    const candidateDigest = buildCandidateDigest(proof, documentHash);
    return (toBase64(<Uint8Array> digest) === toBase64(candidateDigest));
}
```

2. Verwenden Sie zwei `.ts`-Programme (`GetDigest.ts` und `GetRevision.ts`), um die folgenden Schritte auszuführen:

- Fordern Sie einen neuen Digest aus dem Ledger `vehicle-registration` an.
- Fordern Sie einen Nachweis für jede Revision des Dokuments mit VIN `1N4AL11D75C109151` aus der `VehicleRegistration`-Tabelle an.
- Überprüfen Sie die Revisionen mit dem zurückgegebenen Digest und dem Nachweis, indem Sie den Digest neu berechnen.

Das Programm `GetDigest.ts` enthält den folgenden Code.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QLDB } from "aws-sdk";
import { GetDigestRequest, GetDigestResponse } from "aws-sdk/clients/qldb";

import { LEDGER_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { digestResponseToString } from "./qldb/Util";

/**
```

```
 * Get the digest of a ledger's journal.
 * @param ledgerName Name of the ledger to operate on.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with a GetDigestResponse.
 */
export async function getDigestResult(ledgerName: string, qldbClient: QLDB):
 Promise<GetDigestResponse> {
    const request: GetDigestRequest = {
        Name: ledgerName
    };
    const result: GetDigestResponse = await
 qldbClient.getDigest(request).promise();
    return result;
}

/**
 * This is an example for retrieving the digest of a particular ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbClient: QLDB = new QLDB();
        log(`Retrieving the current digest for ledger: ${LEDGER_NAME}.`);
        const digest: GetDigestResponse = await getDigestResult(LEDGER_NAME,
 qldbClient);
        log(`Success. Ledger digest: \n${digestResponseToString(digest)}.`);
    } catch (e) {
        error(`Unable to get a ledger digest: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Verwenden Sie die `getDigest`-Funktion, um einen Digest anzufordern, der die aktuelle Spitze des Journals in Ihrem Ledger abdeckt. Der Tipp des Journals bezieht sich auf den letzten festgeschriebenen Block zu dem Zeitpunkt, zu dem QLDB Ihre Anfrage erhält.

Das Programm `GetRevision.ts` enthält den folgenden Code.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { QLDB } from "aws-sdk";
import { Digest, GetDigestResponse, GetRevisionRequest, GetRevisionResponse,
 ValueHolder } from "aws-sdk/clients/qldb";
import { dom, toBase64 } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { getDigestResult } from './GetDigest';
import { VEHICLE_REGISTRATION } from "./model/SampleData"
import { blockAddressToValueHolder, getMetadataId } from './qldb/BlockAddress';
import { LEDGER_NAME } from './qldb/Constants';
import { error, log } from "./qldb/LogUtil";
import { getBlobValue, valueHolderToString } from "./qldb/Util";
import { flipRandomBit, verifyDocument } from "./qldb/Verifier";
```

```
/**
 * Get the revision data object for a specified document ID and block address.
 * Also returns a proof of the specified revision for verification.
 * @param ledgerName Name of the ledger containing the document to query.
 * @param documentId Unique ID for the document to be verified, contained in the
 committed view of the document.
 * @param blockAddress The location of the block to request.
 * @param digestTipAddress The latest block location covered by the digest.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with a GetRevisionResponse.
 */
async function getRevision(
    ledgerName: string,
    documentId: string,
    blockAddress: ValueHolder,
    digestTipAddress: ValueHolder,
    qldbClient: QLDB
): Promise<GetRevisionResponse> {
    const request: GetRevisionRequest = {
        Name: ledgerName,
        BlockAddress: blockAddress,
        DocumentId: documentId,
        DigestTipAddress: digestTipAddress
    };
    const result: GetRevisionResponse = await
 qldbClient.getRevision(request).promise();
    return result;
}

/**
 * Query the table metadata for a particular vehicle for verification.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin VIN to query the table metadata of a specific registration with.
 * @returns Promise which fulfills with a list of Ion values that contains the
 results of the query.
 */
export async function lookupRegistrationForVin(txn: TransactionExecutor, vin:
 string): Promise<dom.Value[]> {
    log(`Querying the 'VehicleRegistration' table for VIN: ${vin}...`);
    let resultList: dom.Value[];
    const query: string = "SELECT blockAddress, metadata.id FROM
 _ql_committed_VehicleRegistration WHERE data.VIN = ?";

    await txn.execute(query, vin).then(function(result) {
```

```
      resultList = result.getResultList();
    });
    return resultList;
}

/**
 * Verify each version of the registration for the given VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param ledgerName The ledger to get the digest from.
 * @param vin VIN to query the revision history of a specific registration with.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with void.
 * @throws Error: When verification fails.
 */
export async function verifyRegistration(
    txn: TransactionExecutor,
    ledgerName: string,
    vin: string,
    qldbClient: QLDB
): Promise<void> {
    log(`Let's verify the registration with VIN = ${vin}, in ledger =
 ${ledgerName}.`);
    const digest: GetDigestResponse = await getDigestResult(ledgerName,
 qldbClient);
    const digestBytes: Digest = digest.Digest;
    const digestTipAddress: ValueHolder = digest.DigestTipAddress;

    log(
        `Got a ledger digest: digest tip address = \n
${valueHolderToString(digestTipAddress)},
        digest = \n${toBase64(<Uint8Array> digestBytes)}.`
    );
    log(`Querying the registration with VIN = ${vin} to verify each version of the
 registration...`);
    const resultList: dom.Value[] = await lookupRegistrationForVin(txn, vin);
    log("Getting a proof for the document.");

    for (const result of resultList) {
        const blockAddress: ValueHolder =  blockAddressToValueHolder(result);
        const documentId: string = getMetadataId(result);

        const revisionResponse: GetRevisionResponse = await getRevision(
            ledgerName,
            documentId,
```

```
            blockAddress,
            digestTipAddress,
            qldbClient
        );

        const revision: dom.Value = dom.load(revisionResponse.Revision.IonText);
        const documentHash: Uint8Array = getBlobValue(revision, "hash");
        const proof: ValueHolder = revisionResponse.Proof;
        log(`Got back a proof: ${valueHolderToString(proof)}.`);

        let verified: boolean = verifyDocument(documentHash, digestBytes, proof);
        if (!verified) {
           throw new Error("Document revision is not verified.");
        } else {
            log("Success! The document is verified.");
        }
        const alteredDocumentHash: Uint8Array = flipRandomBit(documentHash);

        log(
            `Flipping one bit in the document's hash and assert that the document
 is NOT verified.
            The altered document hash is: ${toBase64(alteredDocumentHash)}`
        );
        verified = verifyDocument(alteredDocumentHash, digestBytes, proof);

        if (verified) {
            throw new Error("Expected altered document hash to not be verified
 against digest.");
        } else {
            log("Success! As expected flipping a bit in the document hash causes
 verification to fail.");
        }
        log(`Finished verifying the registration with VIN = ${vin} in ledger =
 ${ledgerName}.`);
    }
}

/**
 * Verify the integrity of a document revision in a QLDB ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbClient: QLDB = new QLDB();
```

```
        const qldbDriver: QldbDriver = getQldbDriver();

        const registration = VEHICLE_REGISTRATION[0];
        const vin: string = registration.VIN;

        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await verifyRegistration(txn, LEDGER_NAME, vin, qldbClient);
        });
    } catch (e) {
        error(`Unable to verify revision: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Nachdem die `getRevision`-Funktion einen Nachweis für die angegebene Dokumentrevision zurückgibt, verwendet dieses Programm eine clientseitige API, um diese Revision zu überprüfen.

3. Geben Sie den folgenden Befehl ein, um das transpilierte Programm auszuführen.

```
node dist/GetRevision.js
```

Wenn Sie den `vehicle-registration`-Ledger nicht mehr verwenden müssen, fahren Sie mit Schritt 8 (optional): Ressourcen bereinigen fort.

Schritt 8 (optional): Ressourcen bereinigen

Sie können den `vehicle-registration`-Ledger weiterhin verwenden. Wenn Sie ihn nicht mehr benötigen, sollten Sie ihn jedoch löschen.

So löschen Sie den Ledger

1. Verwenden Sie das folgende Programm (`DeleteLedger.ts`), um Ihren `vehicle-registration`-Ledger und all seine Inhalte zu löschen.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { isResourceNotFoundException } from "amazon-qldb-driver-nodejs";
import { AWSError, QLDB } from "aws-sdk";
import { DeleteLedgerRequest, DescribeLedgerRequest } from "aws-sdk/clients/qldb";

import { setDeletionProtection } from "./DeletionProtection";
import { LEDGER_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { sleep } from "./qldb/Util";

const LEDGER_DELETION_POLL_PERIOD_MS = 20000;

/**
 * Send a request to QLDB to delete the specified ledger.
 * @param ledgerName Name of the ledger to be deleted.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with void.
 */
```

```
export async function deleteLedger(ledgerName: string, qldbClient: QLDB):
 Promise<void> {
    log(`Attempting to delete the ledger with name: ${ledgerName}`);
    const request: DeleteLedgerRequest = {
        Name: ledgerName
    };
    await qldbClient.deleteLedger(request).promise();
    log("Success.");
}

/**
 * Wait for the ledger to be deleted.
 * @param ledgerName Name of the ledger to be deleted.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with void.
 */
export async function waitForDeleted(ledgerName: string, qldbClient: QLDB):
 Promise<void> {
    log("Waiting for the ledger to be deleted...");
    const request: DescribeLedgerRequest = {
        Name: ledgerName
    };
    let isDeleted: boolean = false;
    while (true) {
        await qldbClient.describeLedger(request).promise().catch((error: AWSError)
 => {
            if (isResourceNotFoundException(error)) {
                isDeleted = true;
                log("Success. Ledger is deleted.");
            }
        });
        if (isDeleted) {
            break;
        }
        log("The ledger is still being deleted. Please wait...");
        await sleep(LEDGER_DELETION_POLL_PERIOD_MS);
    }
}

/**
 * Delete a ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
```

```
    try {
        const qldbClient: QLDB = new QLDB();
        await setDeletionProtection(LEDGER_NAME, qldbClient, false);
        await deleteLedger(LEDGER_NAME, qldbClient);
        await waitForDeleted(LEDGER_NAME, qldbClient);
    } catch (e) {
        error(`Unable to delete the ledger: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Wenn der Löschschutz für Ihr Ledger aktiviert ist, müssen Sie ihn zuerst deaktivieren, bevor Sie das Ledger mithilfe der QLDB-API löschen können.

2. Geben Sie den folgenden Befehl ein, um das transpilierte Programm auszuführen.

```
node dist/DeleteLedger.js
```

Amazon QLDB-for-Python — Tutorial

In dieser Implementierung der Tutorial-Beispielanwendung verwenden Sie den Amazon QLDB-Treiber mit dem,AWS SDK for Python (Boto3) um ein QLDB-Ledger zu erstellen und es mit Beispieldaten zu füllen.

Während Sie dieses Tutorial durcharbeiten, können Sie sich auf den QLDB-for-PythonAWS SDK for Python (Boto3) beziehen. Informationen zu Transaktionsdatenoperationen finden Sie in der QLDB-Treiber für Python API-Referenz.

> **Note**
>
> Gegebenenfalls enthalten einige Tutorialschritte unterschiedliche Befehle oder Codebeispiele für jede unterstützte Hauptversion des QLDB-Treibers für Python.

Themen

Installation der Amazon QLDB Python-Beispielanwendung

In diesem Abschnitt wird beschrieben, wie die bereitgestellte Amazon QLDB-Beispielanwendung für das step-by-step Python-Tutorial installiert und ausgeführt wird. Der Anwendungsfall für diese Beispielanwendung ist eine Datenbank der Straßenverkehrsbehörde, mit der die vollständigen Verlaufsinformationen über Fahrzeugzulassungen nachverfolgt werden.

Die DMV-Beispielanwendung für Python ist Open Source im GitHub Repository aws-samples/amazon-qldb-dmv-sample -python.

Voraussetzungen

Bevor Sie beginnen, müssen Sie sicherstellen, dass Sie den QLDB-Treiber für Python vervollständigenVoraussetzungen. Dazu gehören die Installation von Python und die folgenden Schritte:

1. Registrieren fürAWS.
2. Erstellen Sie einen Benutzer mit den entsprechenden QLDB-Berechtigungen.
3. Gewähren Sie programmatischen Zugriff für Entwicklungszwecke.

Um alle Schritte in diesem Tutorial abzuschließen, benötigen Sie über die QLDB-API vollen administrativen Zugriff auf Ihre Ledger-Ressource.

Installation

So installieren Sie die Beispielanwendung

1. Geben Sie den folgendenpip Befehl ein, um die Beispielanwendung zu klonen und zu installieren GitHub.

   3.x

   ```
   pip install git+https://github.com/aws-samples/amazon-qldb-dmv-sample-python.git
   ```

   2.x

   ```
   pip install git+https://github.com/aws-samples/amazon-qldb-dmv-sample-
   python.git@v1.0.0
   ```

   Die Beispielanwendung packt den kompletten Quellcode aus diesem Tutorial und seine Abhängigkeiten, einschließlich des Python-Treibers und des AWS SDK for Python (Boto3) zusammen.

2. Bevor Sie den Code an der Befehlszeile ausführen, ändern Sie Ihr aktuelles Arbeitsverzeichnis in den Speicherort, an dem das `pyqldbsamples`-Paket installiert ist. Geben Sie den folgenden Befehl ein.

   ```
   cd $(python -c "import pyqldbsamples; print(pyqldbsamples.__path__[0])")
   ```

3. Fahren Sie mit Schritt 1: Erstellen eines neuen Ledger fort, um das Tutorial zu starten und ein Ledger zu erstellen.

Schritt 1: Erstellen eines neuen Ledger

In diesem Schritt erstellen Sie ein neues Amazon QLDB-Ledger namens`vehicle-registration`.

So erstellen Sie einen neuen Ledger

1. Überprüfen Sie die folgende Datei (`constants.py`), die konstante Werte enthält, die von allen anderen Programmen in diesem Tutorial verwendet werden.

   ```
   # Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
   ```

```
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.


class Constants:
    """
    Constant values used throughout this tutorial.
    """
    LEDGER_NAME = "vehicle-registration"

    VEHICLE_REGISTRATION_TABLE_NAME = "VehicleRegistration"
    VEHICLE_TABLE_NAME = "Vehicle"
    PERSON_TABLE_NAME = "Person"
    DRIVERS_LICENSE_TABLE_NAME = "DriversLicense"

    LICENSE_NUMBER_INDEX_NAME = "LicenseNumber"
    GOV_ID_INDEX_NAME = "GovId"
    VEHICLE_VIN_INDEX_NAME = "VIN"
    LICENSE_PLATE_NUMBER_INDEX_NAME = "LicensePlateNumber"
    PERSON_ID_INDEX_NAME = "PersonId"

    JOURNAL_EXPORT_S3_BUCKET_NAME_PREFIX = "qldb-tutorial-journal-export"
    USER_TABLES = "information_schema.user_tables"
    S3_BUCKET_ARN_TEMPLATE = "arn:aws:s3:::"
    LEDGER_NAME_WITH_TAGS = "tags"
```

```
    RETRY_LIMIT = 4
```

2. Verwenden Sie das folgende Programm (`create_ledger.py`), um einen Ledger mit dem Namen `vehicle-registration` zu erstellen.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO
from time import sleep

from boto3 import client

from pyqldbsamples.constants import Constants

logger = getLogger(__name__)
basicConfig(level=INFO)
qldb_client = client('qldb')

LEDGER_CREATION_POLL_PERIOD_SEC = 10
ACTIVE_STATE = "ACTIVE"
```

```
def create_ledger(name):
    """
    Create a new ledger with the specified name.

    :type name: str
    :param name: Name for the ledger to be created.

    :rtype: dict
    :return: Result from the request.
    """
    logger.info("Let's create the ledger named: {}...".format(name))
    result = qldb_client.create_ledger(Name=name, PermissionsMode='ALLOW_ALL')
    logger.info('Success. Ledger state: {}.'.format(result.get('State')))
    return result


def wait_for_active(name):
    """
    Wait for the newly created ledger to become active.

    :type name: str
    :param name: The ledger to check on.

    :rtype: dict
    :return: Result from the request.
    """
    logger.info('Waiting for ledger to become active...')
    while True:
        result = qldb_client.describe_ledger(Name=name)
        if result.get('State') == ACTIVE_STATE:
            logger.info('Success. Ledger is active and ready to use.')
            return result
        logger.info('The ledger is still creating. Please wait...')
        sleep(LEDGER_CREATION_POLL_PERIOD_SEC)


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Create a ledger and wait for it to be active.
    """
    try:
        create_ledger(ledger_name)
        wait_for_active(ledger_name)
```

```
    except Exception as e:
        logger.exception('Unable to create the ledger!')
        raise e


if __name__ == '__main__':
    main()
```

> **Note**
>
> - Im `create_ledger`-Aufruf müssen Sie einen Ledger-Namen und einen Berechtigungsmodus angeben. Wir empfehlen, den`STANDARD` Berechtigungsmodus, um die Sicherheit Ihrer Ledger-Daten zu maximieren.
> - Wenn Sie einen Ledger erstellen, ist der Löschschutz standardmäßig aktiviert. Dies ist eine Funktion von QLDB, die verhindert, dass Ledger von einem beliebigen Benutzer gelöscht werden. Sie haben die Möglichkeit, den Löschschutz bei der Ledger-Erstellung mithilfe der QLDB-API oder derAWS Command Line Interface (AWS CLI) zu deaktivieren.
> - Optional können Sie auch Tags angeben, die Ihrem Ledger angefügt werden sollen.

3. Geben Sie den folgenden Befehl ein, um das Programm auszuführen.

```
python create_ledger.py
```

Fahren Sie zum Überprüfen der Verbindung mit dem neuen Ledger mit Schritt 2: Testen der Konnektivität des -SDK-for-Python fort.

Schritt 2: Testen der Konnektivität des -SDK-for-Python

In diesem Schritt überprüfen Sie, ob Sie über den API-Endpunkt für Transaktionsdaten eine Verbindung zum`vehicle-registration` Ledger in Amazon QLDB herstellen können.

So testen Sie die Verbindung zum Ledger

1. Verwenden Sie das folgende Programm (`connect_to_ledger.py`), um eine Datensitzungsverbindung mit dem Ledger `vehicle-registration` zu erstellen.

3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from botocore.exceptions import ClientError

from pyqldb.driver.qldb_driver import QldbDriver
from pyqldbsamples.constants import Constants

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_qldb_driver(ledger_name=Constants.LEDGER_NAME, region_name=None,
 endpoint_url=None, boto3_session=None):
    """
    Create a QLDB driver for executing transactions.
```

```
    :type ledger_name: str
    :param ledger_name: The QLDB ledger name.

    :type region_name: str
    :param region_name: See [1].

    :type endpoint_url: str
    :param endpoint_url: See [1].

    :type boto3_session: :py:class:`boto3.session.Session`
    :param boto3_session: The boto3 session to create the client with (see [1]).

    :rtype: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :return: A QLDB driver object.

    [1]: `Boto3 Session.client Reference <https://
boto3.amazonaws.com/v1/documentation/api/latest/reference/core/
session.html#boto3.session.Session.client>`.
    """
    qldb_driver = QldbDriver(ledger_name=ledger_name, region_name=region_name,
 endpoint_url=endpoint_url,
                             boto3_session=boto3_session)
    return qldb_driver


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Connect to a given ledger using default settings.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            logger.info('Listing table names ')
            for table in driver.list_tables():
                logger.info(table)
    except ClientError as ce:
        logger.exception('Unable to list tables.')
        raise ce


if __name__ == '__main__':
    main()
```

> **Note**
>
> - Um Datentransaktionen in Ihrem Ledger auszuführen, müssen Sie ein QLDB-Treiberobjekt erstellen, um eine Verbindung zu einem bestimmten Ledger herzustellen. Dies ist ein anderes Client-Objekt als das `qldb_client`-Objekt, das Sie im vorherigen Schritt zum Erstellen des Ledgers verwendet haben. Dieser vorherige Client wird nur zum Ausführen der Management-API-Operationen verwendet, die in der aufgeführt sindAmazon QLDB API-Referenz.
> - Sie müssen einen Buchnamen angeben, wenn Sie dieses Treiberobjekt erstellen.

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO
```

```
from botocore.exceptions import ClientError

from pyqldb.driver.pooled_qldb_driver import PooledQldbDriver
from pyqldbsamples.constants import Constants

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_qldb_driver(ledger_name=Constants.LEDGER_NAME, region_name=None,
 endpoint_url=None, boto3_session=None):
    """
    Create a QLDB driver for creating sessions.

    :type ledger_name: str
    :param ledger_name: The QLDB ledger name.

    :type region_name: str
    :param region_name: See [1].

    :type endpoint_url: str
    :param endpoint_url: See [1].

    :type boto3_session: :py:class:`boto3.session.Session`
    :param boto3_session: The boto3 session to create the client with (see [1]).

    :rtype: :py:class:`pyqldb.driver.pooled_qldb_driver.PooledQldbDriver`
    :return: A pooled QLDB driver object.

    [1]: `Boto3 Session.client Reference <https://
boto3.amazonaws.com/v1/documentation/api/latest/reference/core/
session.html#boto3.session.Session.client>`.
    """
    qldb_driver = PooledQldbDriver(ledger_name=ledger_name,
 region_name=region_name, endpoint_url=endpoint_url,
                                   boto3_session=boto3_session)
    return qldb_driver


def create_qldb_session():
    """
    Retrieve a QLDB session object.

    :rtype: :py:class:`pyqldb.session.pooled_qldb_session.PooledQldbSession`
```

```
    :return: A pooled QLDB session object.
    """
    qldb_session = pooled_qldb_driver.get_session()
    return qldb_session


pooled_qldb_driver = create_qldb_driver()


if __name__ == '__main__':
    """
    Connect to a session for a given ledger using default settings.
    """
    try:
        qldb_session = create_qldb_session()
        logger.info('Listing table names ')
        for table in qldb_session.list_tables():
            logger.info(table)
    except ClientError:
        logger.exception('Unable to create session.')
```

> **Note**
>
> - Um Datentransaktionen in Ihrem Ledger auszuführen, müssen Sie ein QLDB-Treiberobjekt erstellen, um eine Verbindung zu einem bestimmten Ledger herzustellen. Dies ist ein anderes Client-Objekt als das `qldb_client`-Objekt, das Sie im vorherigen Schritt zum Erstellen des Ledgers verwendet haben. Dieser vorherige Client wird nur zum Ausführen der Management-API-Operationen verwendet, die in der aufgeführt sindAmazon QLDB API-Referenz.
> - Erstellen Sie zunächst ein gepooltes QLDB-Treiberobjekt. Beim Erstellen dieses Treibers müssen Sie einen Ledger-Namen angeben.
> - Anschließend können Sie Sitzungen über dieses Treiberobjekt im Pool erstellen.

2. Geben Sie den folgenden Befehl ein, um das Programm auszuführen.

```
python connect_to_ledger.py
```

Fahren Sie zum Erstellen von Tabellen im `vehicle-registration`-Ledger mit Schritt 3: Tabellen, Indizes und Beispieldaten erstellen fort.

Schritt 3: Tabellen, Indizes und Beispieldaten erstellen

Wenn Ihr Amazon QLDB-Ledger aktiv ist und Verbindungen akzeptiert, können Sie damit beginnen, Tabellen für Daten über Fahrzeuge, deren Besitzer und deren Zulassungsinformationen zu erstellen. Nach dem Erstellen der Tabellen und Indizes können Sie Daten in diese laden.

In diesem Schritt erstellen Sie vier Tabellen im `vehicle-registration`-Ledger:

- `VehicleRegistration`
- `Vehicle`
- `Person`
- `DriversLicense`

Außerdem erzeugen Sie die folgenden Indizes.

Tabellenname	Feld
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Beim Einfügen von Beispieldaten fügen Sie zunächst Dokumente in die `Person`-Tabelle ein. Anschließend verwenden Sie die vom System zugewiesene `id` aus den `Person`-Dokumenten, um die entsprechenden Felder in den relevanten `VehicleRegistration`- und `DriversLicense`-Dokumenten auszufüllen.

> **Tip**
>
> Es hat sich bewährt, den einem Dokument vom System zugewiesenen Schlüssel`id` als Fremdschlüssel zu verwenden. Sie können zwar Felder definieren, die eindeutige Kennungen (z. B. eine Kfz-VIN) sein sollen, die echte eindeutige Kennung eines Dokuments ist aber seine `id`. Dieses Feld ist in den Metadaten des Dokuments enthalten, die Sie in der festgeschriebenen Ansicht (d. h. in der vom System definierten Ansicht der Tabelle) abrufen können.
> Weitere Hinweise zu Ansichten in QLDB finden Sie unterSchlüsselkonzepte. Weitere Informationen zu Metadaten finden Sie unter Metadaten von Dokumenten abfragen.

So erstellen Sie Tabellen und Indizes

1. Verwenden Sie das folgende Programm (`create_table.py`), um die oben genannten Tabellen zu erstellen.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

```
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_table(driver, table_name):
    """
    Create a table with the specified name.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type table_name: str
    :param table_name: Name of the table to create.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Creating the '{}' table...".format(table_name))
    statement = 'CREATE TABLE {}'.format(table_name)
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement))
    logger.info('{} table created successfully.'.format(table_name))
    return len(list(cursor))


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Create registrations, vehicles, owners, and licenses tables.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            create_table(driver, Constants.DRIVERS_LICENSE_TABLE_NAME)
            create_table(driver, Constants.PERSON_TABLE_NAME)
            create_table(driver, Constants.VEHICLE_TABLE_NAME)
            create_table(driver, Constants.VEHICLE_REGISTRATION_TABLE_NAME)
            logger.info('Tables created successfully.')
```

```
    except Exception as e:
        logger.exception('Errors creating tables.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)
```

```
def create_table(transaction_executor, table_name):
    """
    Create a table with the specified name using an Executor object.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type table_name: str
    :param table_name: Name of the table to create.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Creating the '{}' table...".format(table_name))
    statement = 'CREATE TABLE {}'.format(table_name)
    cursor = transaction_executor.execute_statement(statement)
    logger.info('{} table created successfully.'.format(table_name))
    return len(list(cursor))


if __name__ == '__main__':
    """
    Create registrations, vehicles, owners, and licenses tables in a single
 transaction.
    """
    try:
        with create_qldb_session() as session:
            session.execute_lambda(lambda x: create_table(x,
 Constants.DRIVERS_LICENSE_TABLE_NAME) and
                                   create_table(x, Constants.PERSON_TABLE_NAME)
 and
                                   create_table(x, Constants.VEHICLE_TABLE_NAME)
 and
                                   create_table(x,
 Constants.VEHICLE_REGISTRATION_TABLE_NAME),
                                   lambda retry_attempt: logger.info('Retrying
 due to OCC conflict...'))
            logger.info('Tables created successfully.')
    except Exception:
        logger.exception('Errors creating tables.')
```

> **Note**
>
> Dieses Programm zeigt, wie die`execute_lambda` Funktion verwendet wird. In diesem Beispiel führen Sie mehrere `CREATE TABLE`-PartiQL-Anweisungen mit einem einzelnen Lambda-Ausdruck aus.
> Diese Ausführungsfunktion startet implizit eine Transaktion, führt alle Anweisungen im Lambda aus und bestätigt die Transaktion dann automatisch.

2. Geben Sie den folgenden Befehl ein, um das Programm auszuführen.

```
python create_table.py
```

3. Verwenden Sie das folgende Programm (`create_index.py`), um wie zuvor beschrieben Indizes für die Tabellen zu erstellen.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
```

```
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_index(driver, table_name, index_attribute):
    """
    Create an index for a particular table.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type table_name: str
    :param table_name: Name of the table to add indexes for.

    :type index_attribute: str
    :param index_attribute: Index to create on a single attribute.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Creating index on '{}'...".format(index_attribute))
    statement = 'CREATE INDEX on {} ({})'.format(table_name, index_attribute)
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement))
    return len(list(cursor))


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Create indexes on tables in a particular ledger.
    """
    logger.info('Creating indexes on all tables...')
    try:
        with create_qldb_driver(ledger_name) as driver:
            create_index(driver, Constants.PERSON_TABLE_NAME,
 Constants.GOV_ID_INDEX_NAME)
```

```
            create_index(driver, Constants.VEHICLE_TABLE_NAME,
 Constants.VEHICLE_VIN_INDEX_NAME)
            create_index(driver, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 Constants.LICENSE_PLATE_NUMBER_INDEX_NAME)
            create_index(driver, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 Constants.VEHICLE_VIN_INDEX_NAME)
            create_index(driver, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.PERSON_ID_INDEX_NAME)
            create_index(driver, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.LICENSE_NUMBER_INDEX_NAME)
            logger.info('Indexes created successfully.')
    except Exception as e:
        logger.exception('Unable to create indexes.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

```
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_index(transaction_executor, table_name, index_attribute):
    """
    Create an index for a particular table.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type table_name: str
    :param table_name: Name of the table to add indexes for.

    :type index_attribute: str
    :param index_attribute: Index to create on a single attribute.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Creating index on '{}'...".format(index_attribute))
    statement = 'CREATE INDEX on {} ({})'.format(table_name, index_attribute)
    cursor = transaction_executor.execute_statement(statement)
    return len(list(cursor))


if __name__ == '__main__':
    """
    Create indexes on tables in a particular ledger.
    """
    logger.info('Creating indexes on all tables in a single transaction...')
    try:
        with create_qldb_session() as session:
            session.execute_lambda(lambda x: create_index(x,
 Constants.PERSON_TABLE_NAME,
```

```
Constants.GOV_ID_INDEX_NAME)
                                    and create_index(x,
Constants.VEHICLE_TABLE_NAME,

Constants.VEHICLE_VIN_INDEX_NAME)
                                    and create_index(x,
Constants.VEHICLE_REGISTRATION_TABLE_NAME,

Constants.LICENSE_PLATE_NUMBER_INDEX_NAME)
                                    and create_index(x,
Constants.VEHICLE_REGISTRATION_TABLE_NAME,

Constants.VEHICLE_VIN_INDEX_NAME)
                                    and create_index(x,
Constants.DRIVERS_LICENSE_TABLE_NAME,

Constants.PERSON_ID_INDEX_NAME)
                                    and create_index(x,
Constants.DRIVERS_LICENSE_TABLE_NAME,

Constants.LICENSE_NUMBER_INDEX_NAME),
                                    lambda retry_attempt: logger.info('Retrying
due to OCC conflict...'))
           logger.info('Indexes created successfully.')
   except Exception:
       logger.exception('Unable to create indexes.')
```

4. Geben Sie den folgenden Befehl ein, um das Programm auszuführen.

```
python create_index.py
```

So laden Sie Daten in die Tabellen

1. Überprüfen Sie die folgende Datei (`sample_data.py`), die Daten darstellt, die Sie in die `vehicle-registration`-Tabellen einfügen. Diese Datei wird ebenfalls aus dem `amazon.ion`-Paket importiert, um Hilfsfunktionen zum Konvertieren, Analysieren und Drucken von Amazon Ion-Daten bereitzustellen.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
```

```
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
from datetime import datetime
from decimal import Decimal
from logging import basicConfig, getLogger, INFO

from amazon.ion.simple_types import IonPyBool, IonPyBytes, IonPyDecimal, IonPyDict,
 IonPyFloat, IonPyInt, IonPyList, \
    IonPyNull, IonPySymbol, IonPyText, IonPyTimestamp
from amazon.ion.simpleion import dumps, loads

logger = getLogger(__name__)
basicConfig(level=INFO)
IonValue = (IonPyBool, IonPyBytes, IonPyDecimal, IonPyDict, IonPyFloat, IonPyInt,
 IonPyList, IonPyNull, IonPySymbol,
            IonPyText, IonPyTimestamp)


class SampleData:
    """
    Sample domain objects for use throughout this tutorial.
    """
    DRIVERS_LICENSE = [
        {
            'PersonId': '',
            'LicenseNumber': 'LEWISR261LL',
            'LicenseType': 'Learner',
```

```
            'ValidFromDate': datetime(2016, 12, 20),
            'ValidToDate': datetime(2020, 11, 15)
        },
        {
            'PersonId': '',
            'LicenseNumber': 'LOGANB486CG',
            'LicenseType': 'Probationary',
            'ValidFromDate': datetime(2016, 4, 6),
            'ValidToDate': datetime(2020, 11, 15)
        },
        {
            'PersonId': '',
            'LicenseNumber': '744 849 301',
            'LicenseType': 'Full',
            'ValidFromDate': datetime(2017, 12, 6),
            'ValidToDate': datetime(2022, 10, 15)
        },
        {
            'PersonId': '',
            'LicenseNumber': 'P626-168-229-765',
            'LicenseType': 'Learner',
            'ValidFromDate': datetime(2017, 8, 16),
            'ValidToDate': datetime(2021, 11, 15)
        },
        {
            'PersonId': '',
            'LicenseNumber': 'S152-780-97-415-0',
            'LicenseType': 'Probationary',
            'ValidFromDate': datetime(2015, 8, 15),
            'ValidToDate': datetime(2021, 8, 21)
        }
    ]
    PERSON = [
        {
            'FirstName': 'Raul',
            'LastName': 'Lewis',
            'Address': '1719 University Street, Seattle, WA, 98109',
            'DOB': datetime(1963, 8, 19),
            'GovId': 'LEWISR261LL',
            'GovIdType': 'Driver License'
        },
        {
            'FirstName': 'Brent',
            'LastName': 'Logan',
```

```
            'DOB': datetime(1967, 7, 3),
            'Address': '43 Stockert Hollow Road, Everett, WA, 98203',
            'GovId': 'LOGANB486CG',
            'GovIdType': 'Driver License'
        },
        {
            'FirstName': 'Alexis',
            'LastName': 'Pena',
            'DOB': datetime(1974, 2, 10),
            'Address': '4058 Melrose Street, Spokane Valley, WA, 99206',
            'GovId': '744 849 301',
            'GovIdType': 'SSN'
        },
        {
            'FirstName': 'Melvin',
            'LastName': 'Parker',
            'DOB': datetime(1976, 5, 22),
            'Address': '4362 Ryder Avenue, Seattle, WA, 98101',
            'GovId': 'P626-168-229-765',
            'GovIdType': 'Passport'
        },
        {
            'FirstName': 'Salvatore',
            'LastName': 'Spencer',
            'DOB': datetime(1997, 11, 15),
            'Address': '4450 Honeysuckle Lane, Seattle, WA, 98101',
            'GovId': 'S152-780-97-415-0',
            'GovIdType': 'Passport'
        }
    ]
    VEHICLE = [
        {
            'VIN': '1N4AL11D75C109151',
            'Type': 'Sedan',
            'Year': 2011,
            'Make': 'Audi',
            'Model': 'A5',
            'Color': 'Silver'
        },
        {
            'VIN': 'KM8SRDHF6EU074761',
            'Type': 'Sedan',
            'Year': 2015,
            'Make': 'Tesla',
```

```
            'Model': 'Model S',
            'Color': 'Blue'
        },
        {
            'VIN': '3HGGK5G53FM761765',
            'Type': 'Motorcycle',
            'Year': 2011,
            'Make': 'Ducati',
            'Model': 'Monster 1200',
            'Color': 'Yellow'
        },
        {
            'VIN': '1HVBBAANXWH544237',
            'Type': 'Semi',
            'Year': 2009,
            'Make': 'Ford',
            'Model': 'F 150',
            'Color': 'Black'
        },
        {
            'VIN': '1C4RJFAG0FC625797',
            'Type': 'Sedan',
            'Year': 2019,
            'Make': 'Mercedes',
            'Model': 'CLK 350',
            'Color': 'White'
        }
    ]
    VEHICLE_REGISTRATION = [
        {
            'VIN': '1N4AL11D75C109151',
            'LicensePlateNumber': 'LEWISR261LL',
            'State': 'WA',
            'City': 'Seattle',
            'ValidFromDate': datetime(2017, 8, 21),
            'ValidToDate': datetime(2020, 5, 11),
            'PendingPenaltyTicketAmount': Decimal('90.25'),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        },
        {
            'VIN': 'KM8SRDHF6EU074761',
```

```
            'LicensePlateNumber': 'CA762X',
            'State': 'WA',
            'City': 'Kent',
            'PendingPenaltyTicketAmount': Decimal('130.75'),
            'ValidFromDate': datetime(2017, 9, 14),
            'ValidToDate': datetime(2020, 6, 25),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        },
        {
            'VIN': '3HGGK5G53FM761765',
            'LicensePlateNumber': 'CD820Z',
            'State': 'WA',
            'City': 'Everett',
            'PendingPenaltyTicketAmount': Decimal('442.30'),
            'ValidFromDate': datetime(2011, 3, 17),
            'ValidToDate': datetime(2021, 3, 24),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        },
        {
            'VIN': '1HVBBAANXWH544237',
            'LicensePlateNumber': 'LS477D',
            'State': 'WA',
            'City': 'Tacoma',
            'PendingPenaltyTicketAmount': Decimal('42.20'),
            'ValidFromDate': datetime(2011, 10, 26),
            'ValidToDate': datetime(2023, 9, 25),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        },
        {
            'VIN': '1C4RJFAG0FC625797',
            'LicensePlateNumber': 'TH393F',
            'State': 'WA',
            'City': 'Olympia',
            'PendingPenaltyTicketAmount': Decimal('30.45'),
            'ValidFromDate': datetime(2013, 9, 2),
```

```
            'ValidToDate': datetime(2024, 3, 19),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        }
    ]


def convert_object_to_ion(py_object):
    """
    Convert a Python object into an Ion object.

    :type py_object: object
    :param py_object: The object to convert.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyValue`
    :return: The converted Ion object.
    """
    ion_object = loads(dumps(py_object))
    return ion_object


def to_ion_struct(key, value):
    """
    Convert the given key and value into an Ion struct.

    :type key: str
    :param key: The key which serves as an unique identifier.

    :type value: str
    :param value: The value associated with a given key.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The Ion dictionary object.
    """
    ion_struct = dict()
    ion_struct[key] = value
    return loads(str(ion_struct))


def get_document_ids(transaction_executor, table_name, field, value):
    """
    Gets the document IDs from the given table.
```

```
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type table_name: str
    :param table_name: The table name to query.

    :type field: str
    :param field: A field to query.

    :type value: str
    :param value: The key of the given field.

    :rtype: list
    :return: A list of document IDs.
    """
    query = "SELECT id FROM {} AS t BY id WHERE t.{} = ?".format(table_name, field)
    cursor = transaction_executor.execute_statement(query,
 convert_object_to_ion(value))
    return list(map(lambda table: table.get('id'), cursor))


def get_document_ids_from_dml_results(result):
    """
    Return a list of modified document IDs as strings from DML results.

    :type result: :py:class:`pyqldb.cursor.buffered_cursor.BufferedCursor`
    :param: result: The result set from DML operation.

    :rtype: list
    :return: List of document IDs.
    """
    ret_val = list(map(lambda x: x.get('documentId'), result))
    return ret_val


def print_result(cursor):
    """
    Pretty print the result set. Returns the number of documents in the result set.

    :type cursor: :py:class:`pyqldb.cursor.stream_cursor.StreamCursor`/
                  :py:class:`pyqldb.cursor.buffered_cursor.BufferedCursor`
    :param cursor: An instance of the StreamCursor or BufferedCursor class.
```

```
    :rtype: int
    :return: Number of documents in the result set.
    """
    result_counter = 0
    for row in cursor:
        # Each row would be in Ion format.
        print_ion(row)
        result_counter += 1
    return result_counter


def print_ion(ion_value):
    """
    Pretty print an Ion Value.

    :type ion_value: :py:class:`amazon.ion.simple_types.IonPySymbol`
    :param ion_value: Any Ion Value to be pretty printed.
    """
    logger.info(dumps(ion_value, binary=False, indent='  ',
 omit_version_marker=True))
```

> **Note**
>
> Die `get_document_ids`-Funktion führt eine Abfrage aus, die vom System zugewiesene Dokument-IDs aus einer Tabelle zurückgibt. Weitere Informationen hierzu finden Sie unter Verwenden der BY-Klausel zur Abfrage der Dokument-ID.

2. Verwenden Sie das folgende Programm (`insert_document.py`), um Beispieldaten in Ihre Tabellen einzufügen.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
```

```
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.model.sample_data import convert_object_to_ion, SampleData,
 get_document_ids_from_dml_results
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def update_person_id(document_ids):
    """
    Update the PersonId value for DriversLicense records and the PrimaryOwner
 value for VehicleRegistration records.

    :type document_ids: list
    :param document_ids: List of document IDs.

    :rtype: list
    :return: Lists of updated DriversLicense records and updated
 VehicleRegistration records.
    """
    new_drivers_licenses = SampleData.DRIVERS_LICENSE.copy()
    new_vehicle_registrations = SampleData.VEHICLE_REGISTRATION.copy()
    for i in range(len(SampleData.PERSON)):
        drivers_license = new_drivers_licenses[i]
        registration = new_vehicle_registrations[i]
```

```
        drivers_license.update({'PersonId': str(document_ids[i])})
        registration['Owners']['PrimaryOwner'].update({'PersonId':
 str(document_ids[i])})
    return new_drivers_licenses, new_vehicle_registrations


def insert_documents(driver, table_name, documents):
    """
    Insert the given list of documents into a table in a single transaction.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type table_name: str
    :param table_name: Name of the table to insert documents into.

    :type documents: list
    :param documents: List of documents to insert.

    :rtype: list
    :return: List of documents IDs for the newly inserted documents.
    """
    logger.info('Inserting some documents in the {}
 table...'.format(table_name))
    statement = 'INSERT INTO {} ?'.format(table_name)
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement,

 convert_object_to_ion(documents)))
    list_of_document_ids = get_document_ids_from_dml_results(cursor)

    return list_of_document_ids


def update_and_insert_documents(driver):
    """
    Handle the insertion of documents and updating PersonIds.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.
    """
    list_ids = insert_documents(driver, Constants.PERSON_TABLE_NAME,
 SampleData.PERSON)
```

```
    logger.info("Updating PersonIds for 'DriversLicense' and PrimaryOwner for
 'VehicleRegistration'...")
    new_licenses, new_registrations = update_person_id(list_ids)

    insert_documents(driver, Constants.VEHICLE_TABLE_NAME, SampleData.VEHICLE)
    insert_documents(driver, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 new_registrations)
    insert_documents(driver, Constants.DRIVERS_LICENSE_TABLE_NAME, new_licenses)


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Insert documents into a table in a QLDB ledger.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            # An INSERT statement creates the initial revision of a document
 with a version number of zero.
            # QLDB also assigns a unique document identifier in GUID format as
 part of the metadata.
            update_and_insert_documents(driver)
            logger.info('Documents inserted successfully!')
    except Exception as e:
        logger.exception('Error inserting or updating documents.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
```

```
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.model.sample_data import convert_object_to_ion, SampleData,
 get_document_ids_from_dml_results
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def update_person_id(document_ids):
    """
    Update the PersonId value for DriversLicense records and the PrimaryOwner
 value for VehicleRegistration records.

    :type document_ids: list
    :param document_ids: List of document IDs.

    :rtype: list
    :return: Lists of updated DriversLicense records and updated
 VehicleRegistration records.
    """
    new_drivers_licenses = SampleData.DRIVERS_LICENSE.copy()
    new_vehicle_registrations = SampleData.VEHICLE_REGISTRATION.copy()
    for i in range(len(SampleData.PERSON)):
        drivers_license = new_drivers_licenses[i]
        registration = new_vehicle_registrations[i]
        drivers_license.update({'PersonId': str(document_ids[i])})
```

```
        registration['Owners']['PrimaryOwner'].update({'PersonId':
 str(document_ids[i])})
    return new_drivers_licenses, new_vehicle_registrations


def insert_documents(transaction_executor, table_name, documents):
    """
    Insert the given list of documents into a table in a single transaction.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type table_name: str
    :param table_name: Name of the table to insert documents into.

    :type documents: list
    :param documents: List of documents to insert.

    :rtype: list
    :return: List of documents IDs for the newly inserted documents.
    """
    logger.info('Inserting some documents in the {}
 table...'.format(table_name))
    statement = 'INSERT INTO {} ?'.format(table_name)
    cursor = transaction_executor.execute_statement(statement,
 convert_object_to_ion(documents))
    list_of_document_ids = get_document_ids_from_dml_results(cursor)

    return list_of_document_ids


def update_and_insert_documents(transaction_executor):
    """
    Handle the insertion of documents and updating PersonIds all in a single
 transaction.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    """
    list_ids = insert_documents(transaction_executor,
 Constants.PERSON_TABLE_NAME, SampleData.PERSON)
```

```
    logger.info("Updating PersonIds for 'DriversLicense' and PrimaryOwner for
 'VehicleRegistration'...")
    new_licenses, new_registrations = update_person_id(list_ids)

    insert_documents(transaction_executor, Constants.VEHICLE_TABLE_NAME,
 SampleData.VEHICLE)
    insert_documents(transaction_executor,
 Constants.VEHICLE_REGISTRATION_TABLE_NAME, new_registrations)
    insert_documents(transaction_executor, Constants.DRIVERS_LICENSE_TABLE_NAME,
 new_licenses)


if __name__ == '__main__':
    """
    Insert documents into a table in a QLDB ledger.
    """
    try:
        with create_qldb_session() as session:
            # An INSERT statement creates the initial revision of a document
 with a version number of zero.
            # QLDB also assigns a unique document identifier in GUID format as
 part of the metadata.
            session.execute_lambda(lambda executor:
 update_and_insert_documents(executor),
                                   lambda retry_attempt: logger.info('Retrying
 due to OCC conflict...'))
            logger.info('Documents inserted successfully!')
    except Exception:
        logger.exception('Error inserting or updating documents.')
```

> **Note**
>
> - Dieses Programm veranschaulicht, wie die `execute_statement`-Funktion mit parametrisierten Werten aufgerufen wird. Zusätzlich zu der PartiQL-Anweisung, die Sie ausführen möchten, können Sie Datenparameter übergeben. Verwenden Sie ein Fragezeichen (?) als Variablenplatzhalter in Ihrer Anweisungszeichenfolge.
> - Wenn eine `INSERT`-Anweisung erfolgreich ist, wird die `id` jedes eingefügten Dokuments zurückgegeben.

3. Geben Sie den folgenden Befehl ein, um das Programm auszuführen.

```
python insert_document.py
```

Als Nächstes können Sie mit SELECT-Anweisungen Daten aus den Tabellen im `vehicle-registration`-Ledger lesen. Fahren Sie mit Schritt 4: Abfragen der Tabellen in einem Ledger fort.

Schritt 4: Abfragen der Tabellen in einem Ledger

Nachdem Sie Tabellen in einem Amazon QLDB-Ledger erstellt und sie mit Daten geladen haben, können Sie Abfragen ausführen, um die Fahrzeugregistrierungsdaten zu überprüfen, die Sie gerade eingegeben haben. QLDB verwendet PartiQL als Abfragesprache und Amazon Ion als dokumentorientiertes Datenmodell.

PartiQL ist eine quelloffene, SQL-kompatible Abfragesprache, die für die Arbeit mit Ion erweitert wurde. Mit PartiSQL können Sie Ihre Daten mit vertrauten SQL-Operatoren einfügen, abfragen und verwalten. Amazon Ion ist eine Obermenge von JSON. Ion ist ein dokumentenbasiertes Open-Source-Datenformat, das Ihnen die Flexibilität bietet, strukturierte, halbstrukturierte und verschachtelte Daten zu speichern und zu verarbeiten.

In diesem Schritt verwenden Sie SELECT-Anweisungen zum Lesen von Daten aus den Tabellen im `vehicle-registration`-Ledger.

> **Warning**
>
> Wenn Sie eine Abfrage in QLDB ohne indizierte Suche ausführen, wird ein vollständiger Tabellenscan aufgerufen. PartiQL unterstützt solche Abfragen, da es SQL-kompatibel ist. Führen Sie jedoch keine Tabellenscans für produktive Anwendungsfälle in QLDB aus. Tabellenscans können bei großen Tabellen zu Leistungsproblemen führen, einschließlich Parallelitätskonflikten und Transaktions-Timeouts.
>
> Um Tabellenscans zu vermeiden, müssen Sie Anweisungen mit einer`WHERE` Prädikatklausel ausführen, die einen Gleichheitsoperator für ein indiziertes Feld oder eine Dokument-ID verwenden, z. B.`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`. Weitere Informationen finden Sie unter Optimieren der Abfrageleistung.

So fragen Sie die Tabellen ab

1. Verwenden Sie das folgende Programm (`find_vehicles.py`), um alle Fahrzeuge abzufragen, die für eine Person in Ihrem Ledger zugelassen sind.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import get_document_ids, print_result,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)
```

```
def find_vehicles_for_owner(driver, gov_id):
    """
    Find vehicles registered under a driver using their government ID.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type gov_id: str
    :param gov_id: The owner's government ID.
    """
    document_ids = driver.execute_lambda(lambda executor:
 get_document_ids(executor, Constants.PERSON_TABLE_NAME,

 'GovId', gov_id))

    query = "SELECT Vehicle FROM Vehicle INNER JOIN VehicleRegistration AS r " \
            "ON Vehicle.VIN = r.VIN WHERE r.Owners.PrimaryOwner.PersonId = ?"

    for ids in document_ids:
        cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(query, ids))
        logger.info('List of Vehicles for owner with GovId:
 {}...'.format(gov_id))
        print_result(cursor)


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Find all vehicles registered under a person.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            # Find all vehicles registered under a person.
            gov_id = SampleData.PERSON[0]['GovId']
            find_vehicles_for_owner(driver, gov_id)
    except Exception as e:
        logger.exception('Error getting vehicles for owner.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import get_document_ids, print_result,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def find_vehicles_for_owner(transaction_executor, gov_id):
    """
    Find vehicles registered under a driver using their government ID.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
```

```
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type gov_id: str
    :param gov_id: The owner's government ID.
    """
    document_ids = get_document_ids(transaction_executor,
 Constants.PERSON_TABLE_NAME, 'GovId', gov_id)

    query = "SELECT Vehicle FROM Vehicle INNER JOIN VehicleRegistration AS r " \
            "ON Vehicle.VIN = r.VIN WHERE r.Owners.PrimaryOwner.PersonId = ?"

    for ids in document_ids:
        cursor = transaction_executor.execute_statement(query, ids)
        logger.info('List of Vehicles for owner with GovId:
 {}...'.format(gov_id))
        print_result(cursor)


if __name__ == '__main__':
    """
    Find all vehicles registered under a person.
    """
    try:
        with create_qldb_session() as session:
            # Find all vehicles registered under a person.
            gov_id = SampleData.PERSON[0]['GovId']
            session.execute_lambda(lambda executor:
 find_vehicles_for_owner(executor, gov_id),
                                   lambda retry_attempt: logger.info('Retrying
 due to OCC conflict...'))
    except Exception:
        logger.exception('Error getting vehicles for owner.')
```

> **Note**
>
> Zunächst fragt dieses Programm die `Person`-Tabelle nach dem Dokument mit `GovId` `LEWISR261LL` ab, um sein `id`-Metadatenfeld abzurufen.
> Anschließend verwendet es dieses Dokument `id` als Fremdschlüssel, um die `VehicleRegistration`-Tabelle nach `PrimaryOwner.PersonId` abzufragen. Es

tritt führt außerdem `VehicleRegistration` mit der `Vehicle`-Tabelle im `VIN`-Feld zusammen.

2. Geben Sie den folgenden Befehl ein, um das Programm auszuführen.

```
python find_vehicles.py
```

Weitere Informationen zum Ändern von Dokumenten in den Tabellen im `vehicle-registration`-Ledger finden Sie unter Schritt 5: Dokumente in einem Ledger ändern.

Schritt 5: Dokumente in einem Ledger ändern

Da Sie nun über Daten verfügen, mit denen Sie arbeiten können, können Sie damit beginnen, Änderungen an Dokumenten im`vehicle-registration` Hauptbuch in Amazon QLDB vorzunehmen. In diesem Schritt verdeutlichen die folgenden Codebeispiele, wie Sie Data Manipulation Language (DML)-Anweisungen ausführen. Diese Anweisungen aktualisieren den primären Eigentümer eines Fahrzeugs und fügen einem anderen Fahrzeug einen sekundären Eigentümer hinzu.

So ändern Sie Dokumente

1. Verwenden Sie das folgende Programm (`transfer_vehicle_ownership.py`), um den primären Besitzer des Fahrzeugs mit VIN `1N4AL11D75C109151` in Ihrem Ledger zu aktualisieren.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
```

```
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.add_secondary_owner import get_document_ids, print_result,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.model.sample_data import convert_object_to_ion
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def find_person_from_document_id(transaction_executor, document_id):
    """
    Query a driver's information using the given ID.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type document_id: :py:class:`amazon.ion.simple_types.IonPyText`
    :param document_id: The document ID required to query for the person.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The resulting document from the query.
    """
    query = 'SELECT p.* FROM Person AS p BY pid WHERE pid = ?'
    cursor = transaction_executor.execute_statement(query, document_id)
    return next(cursor)


def find_primary_owner_for_vehicle(driver, vin):
```

```
    """
    Find the primary owner of a vehicle given its VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: The VIN to find primary owner for.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The resulting document from the query.
    """
    logger.info('Finding primary owner for vehicle with VIN: {}.'.format(vin))
    query = "SELECT Owners.PrimaryOwner.PersonId FROM VehicleRegistration AS v
 WHERE v.VIN = ?"
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(query, convert_object_to_ion(vin)))
    try:
        return driver.execute_lambda(lambda executor:
 find_person_from_document_id(executor,

   next(cursor).get('PersonId')))
    except StopIteration:
        logger.error('No primary owner registered for this vehicle.')
        return None


def update_vehicle_registration(driver, vin, document_id):
    """
    Update the primary owner for a vehicle using the given VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: The VIN for the vehicle to operate on.

    :type document_id: :py:class:`amazon.ion.simple_types.IonPyText`
    :param document_id: New PersonId for the primary owner.

    :raises RuntimeError: If no vehicle registration was found using the given
 document ID and VIN.
    """
```

```
    logger.info('Updating the primary owner for vehicle with Vin:
 {}...'.format(vin))
    statement = "UPDATE VehicleRegistration AS r SET
 r.Owners.PrimaryOwner.PersonId = ? WHERE r.VIN = ?"
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement, document_id,

 convert_object_to_ion(vin)))
    try:
        print_result(cursor)
        logger.info('Successfully transferred vehicle with VIN: {} to new
 owner.'.format(vin))
    except StopIteration:
        raise RuntimeError('Unable to transfer vehicle, could not find
 registration.')


def validate_and_update_registration(driver, vin, current_owner, new_owner):
    """
    Validate the current owner of the given vehicle and transfer its ownership
 to a new owner.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: The VIN of the vehicle to transfer ownership of.

    :type current_owner: str
    :param current_owner: The GovId of the current owner of the vehicle.

    :type new_owner: str
    :param new_owner: The GovId of the new owner of the vehicle.

    :raises RuntimeError: If unable to verify primary owner.
    """
    primary_owner = find_primary_owner_for_vehicle(driver, vin)
    if primary_owner is None or primary_owner['GovId'] != current_owner:
        raise RuntimeError('Incorrect primary owner identified for vehicle,
 unable to transfer.')

    document_ids = driver.execute_lambda(lambda executor:
 get_document_ids(executor, Constants.PERSON_TABLE_NAME,
```

```
 'GovId', new_owner))
    update_vehicle_registration(driver, vin, document_ids[0])


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Find primary owner for a particular vehicle's VIN.
    Transfer to another primary owner for a particular vehicle's VIN.
    """
    vehicle_vin = SampleData.VEHICLE[0]['VIN']
    previous_owner = SampleData.PERSON[0]['GovId']
    new_owner = SampleData.PERSON[1]['GovId']

    try:
        with create_qldb_driver(ledger_name) as driver:
            validate_and_update_registration(driver, vehicle_vin,
 previous_owner, new_owner)
    except Exception as e:
        logger.exception('Error updating VehicleRegistration.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
```

```
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.add_secondary_owner import get_document_ids, print_result,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.model.sample_data import convert_object_to_ion
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def find_person_from_document_id(transaction_executor, document_id):
    """
    Query a driver's information using the given ID.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type document_id: :py:class:`amazon.ion.simple_types.IonPyText`
    :param document_id: The document ID required to query for the person.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The resulting document from the query.
    """
    query = 'SELECT p.* FROM Person AS p BY pid WHERE pid = ?'
    cursor = transaction_executor.execute_statement(query, document_id)
    return next(cursor)


def find_primary_owner_for_vehicle(transaction_executor, vin):
    """
    Find the primary owner of a vehicle given its VIN.
```

```
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type vin: str
    :param vin: The VIN to find primary owner for.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The resulting document from the query.
    """
    logger.info('Finding primary owner for vehicle with VIN: {}.'.format(vin))
    query = "SELECT Owners.PrimaryOwner.PersonId FROM VehicleRegistration AS v
 WHERE v.VIN = ?"
    cursor = transaction_executor.execute_statement(query,
 convert_object_to_ion(vin))
    try:
        return find_person_from_document_id(transaction_executor,
 next(cursor).get('PersonId'))
    except StopIteration:
        logger.error('No primary owner registered for this vehicle.')
        return None


def update_vehicle_registration(transaction_executor, vin, document_id):
    """
    Update the primary owner for a vehicle using the given VIN.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type vin: str
    :param vin: The VIN for the vehicle to operate on.

    :type document_id: :py:class:`amazon.ion.simple_types.IonPyText`
    :param document_id: New PersonId for the primary owner.

    :raises RuntimeError: If no vehicle registration was found using the given
 document ID and VIN.
    """
    logger.info('Updating the primary owner for vehicle with Vin:
 {}...'.format(vin))
```

```
    statement = "UPDATE VehicleRegistration AS r SET
 r.Owners.PrimaryOwner.PersonId = ? WHERE r.VIN = ?"
    cursor = transaction_executor.execute_statement(statement, document_id,
 convert_object_to_ion(vin))
    try:
        print_result(cursor)
        logger.info('Successfully transferred vehicle with VIN: {} to new
 owner.'.format(vin))
    except StopIteration:
        raise RuntimeError('Unable to transfer vehicle, could not find
 registration.')


def validate_and_update_registration(transaction_executor, vin, current_owner,
 new_owner):
    """
    Validate the current owner of the given vehicle and transfer its ownership
 to a new owner in a single transaction.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type vin: str
    :param vin: The VIN of the vehicle to transfer ownership of.

    :type current_owner: str
    :param current_owner: The GovId of the current owner of the vehicle.

    :type new_owner: str
    :param new_owner: The GovId of the new owner of the vehicle.

    :raises RuntimeError: If unable to verify primary owner.
    """
    primary_owner = find_primary_owner_for_vehicle(transaction_executor, vin)
    if primary_owner is None or primary_owner['GovId'] != current_owner:
        raise RuntimeError('Incorrect primary owner identified for vehicle,
 unable to transfer.')

    document_id = next(get_document_ids(transaction_executor,
 Constants.PERSON_TABLE_NAME, 'GovId', new_owner))

    update_vehicle_registration(transaction_executor, vin, document_id)
```

```
if __name__ == '__main__':
    """
    Find primary owner for a particular vehicle's VIN.
    Transfer to another primary owner for a particular vehicle's VIN.
    """
    vehicle_vin = SampleData.VEHICLE[0]['VIN']
    previous_owner = SampleData.PERSON[0]['GovId']
    new_owner = SampleData.PERSON[1]['GovId']

    try:
        with create_qldb_session() as session:
            session.execute_lambda(lambda executor:
 validate_and_update_registration(executor, vehicle_vin,

     previous_owner, new_owner),
                                   retry_indicator=lambda retry_attempt:
 logger.info('Retrying due to OCC conflict...'))
    except Exception:
        logger.exception('Error updating VehicleRegistration.')
```

2. Geben Sie den folgenden Befehl ein, um das Programm auszuführen.

```
python transfer_vehicle_ownership.py
```

3. Verwenden Sie das folgende Programm (`add_secondary_owner.py`), um dem Fahrzeug mit VIN `KM8SRDHF6EU074761` in Ihrem Ledger einen sekundären Besitzer hinzuzufügen.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
```

```
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import to_ion_struct, get_document_ids,
 print_result, SampleData, \
    convert_object_to_ion
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def get_document_id_by_gov_id(driver, government_id):
    """
    Find a driver's person ID using the given government ID.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type government_id: str
    :param government_id: A driver's government ID.

    :rtype: list
    :return: A list of document IDs.
    """
    logger.info("Finding secondary owner's person ID using given government ID:
 {}.".format(government_id))
    return driver.execute_lambda(lambda executor: get_document_ids(executor,
 Constants.PERSON_TABLE_NAME, 'GovId',

 government_id))
```

```
def is_secondary_owner_for_vehicle(driver, vin, secondary_owner_id):
    """
    Check whether a secondary owner has already been registered for the given
 VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: VIN of the vehicle to query.

    :type secondary_owner_id: str
    :param secondary_owner_id: The secondary owner's person ID.

    :rtype: bool
    :return: If the driver has already been registered.
    """
    logger.info('Finding secondary owners for vehicle with VIN:
 {}...'.format(vin))
    query = 'SELECT Owners.SecondaryOwners FROM VehicleRegistration AS v WHERE
 v.VIN = ?'
    rows = driver.execute_lambda(lambda executor:
 executor.execute_statement(query, convert_object_to_ion(vin)))

    for row in rows:
        secondary_owners = row.get('SecondaryOwners')
        person_ids = map(lambda owner: owner.get('PersonId').text,
 secondary_owners)
        if secondary_owner_id in person_ids:
            return True
    return False


def add_secondary_owner_for_vin(driver, vin, parameter):
    """
    Add a secondary owner into `VehicleRegistration` table for a particular VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: VIN of the vehicle to add a secondary owner for.
```

```
    :type parameter: :py:class:`amazon.ion.simple_types.IonPyValue`
    :param parameter: The Ion value or Python native type that is convertible to
 Ion for filling in parameters of the
                      statement.
    """
    logger.info('Inserting secondary owner for vehicle with VIN:
 {}...'.format(vin))
    statement = "FROM VehicleRegistration AS v WHERE v.VIN = ? INSERT INTO
 v.Owners.SecondaryOwners VALUE ?"

    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement, convert_object_to_ion(vin),

 parameter))
    logger.info('VehicleRegistration Document IDs which had secondary owners
 added: ')
    print_result(cursor)


def register_secondary_owner(driver, vin, gov_id):
    """
    Register a secondary owner for a vehicle if they are not already registered.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: VIN of the vehicle to register a secondary owner for.

    :type gov_id: str
    :param gov_id: The government ID of the owner.
    """
    logger.info('Finding the secondary owners for vehicle with VIN:
 {}.'.format(vin))

    document_ids = get_document_id_by_gov_id(driver, gov_id)

    for document_id in document_ids:
        if is_secondary_owner_for_vehicle(driver, vin, document_id):
            logger.info('Person with ID {} has already been added as a secondary
 owner of this vehicle.'.format(gov_id))
        else:
            add_secondary_owner_for_vin(driver, vin, to_ion_struct('PersonId',
 document_id))
```

```
def main(ledger_name=Constants.LEDGER_NAME):
    """
    Finds and adds secondary owners for a vehicle.
    """
    vin = SampleData.VEHICLE[1]['VIN']
    gov_id = SampleData.PERSON[0]['GovId']
    try:
        with create_qldb_driver(ledger_name) as driver:
            register_secondary_owner(driver, vin, gov_id)
            logger.info('Secondary owners successfully updated.')
    except Exception as e:
        logger.exception('Error adding secondary owner.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
```

```
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import to_ion_struct, get_document_ids,
 print_result, SampleData, \
    convert_object_to_ion
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def get_document_id_by_gov_id(transaction_executor, government_id):
    """
    Find a driver's person ID using the given government ID.
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    :type government_id: str
    :param government_id: A driver's government ID.
    :rtype: list
    :return: A list of document IDs.
    """
    logger.info("Finding secondary owner's person ID using given government ID:
 {}.".format(government_id))
    return get_document_ids(transaction_executor, Constants.PERSON_TABLE_NAME,
 'GovId', government_id)


def is_secondary_owner_for_vehicle(transaction_executor, vin,
 secondary_owner_id):
    """
    Check whether a secondary owner has already been registered for the given
 VIN.
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    :type vin: str
    :param vin: VIN of the vehicle to query.
    :type secondary_owner_id: str
```

```
    :param secondary_owner_id: The secondary owner's person ID.
    :rtype: bool
    :return: If the driver has already been registered.
    """
    logger.info('Finding secondary owners for vehicle with VIN:
 {}...'.format(vin))
    query = 'SELECT Owners.SecondaryOwners FROM VehicleRegistration AS v WHERE
 v.VIN = ?'
    rows = transaction_executor.execute_statement(query,
 convert_object_to_ion(vin))

    for row in rows:
        secondary_owners = row.get('SecondaryOwners')
        person_ids = map(lambda owner: owner.get('PersonId').text,
 secondary_owners)
        if secondary_owner_id in person_ids:
            return True
    return False


def add_secondary_owner_for_vin(transaction_executor, vin, parameter):
    """
    Add a secondary owner into `VehicleRegistration` table for a particular VIN.
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    :type vin: str
    :param vin: VIN of the vehicle to add a secondary owner for.
    :type parameter: :py:class:`amazon.ion.simple_types.IonPyValue`
    :param parameter: The Ion value or Python native type that is convertible to
 Ion for filling in parameters of the
                      statement.
    """
    logger.info('Inserting secondary owner for vehicle with VIN:
 {}...'.format(vin))
    statement = "FROM VehicleRegistration AS v WHERE v.VIN = '{}' INSERT INTO
 v.Owners.SecondaryOwners VALUE ?"\
        .format(vin)

    cursor = transaction_executor.execute_statement(statement, parameter)
    logger.info('VehicleRegistration Document IDs which had secondary owners
 added: ')
    print_result(cursor)
```

```
def register_secondary_owner(transaction_executor, vin, gov_id):
    """
    Register a secondary owner for a vehicle if they are not already registered.
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    :type vin: str
    :param vin: VIN of the vehicle to register a secondary owner for.
    :type gov_id: str
    :param gov_id: The government ID of the owner.
    """
    logger.info('Finding the secondary owners for vehicle with VIN:
 {}.'.format(vin))
    document_ids = get_document_id_by_gov_id(transaction_executor, gov_id)

    for document_id in document_ids:
        if is_secondary_owner_for_vehicle(transaction_executor, vin,
 document_id):
            logger.info('Person with ID {} has already been added as a secondary
 owner of this vehicle.'.format(gov_id))
        else:
            add_secondary_owner_for_vin(transaction_executor, vin,
 to_ion_struct('PersonId', document_id))


if __name__ == '__main__':
    """
    Finds and adds secondary owners for a vehicle.
    """
    vin = SampleData.VEHICLE[1]['VIN']
    gov_id = SampleData.PERSON[0]['GovId']
    try:
        with create_qldb_session() as session:
            session.execute_lambda(lambda executor:
 register_secondary_owner(executor, vin, gov_id),
                                   lambda retry_attempt: logger.info('Retrying
 due to OCC conflict...'))
            logger.info('Secondary owners successfully updated.')
    except Exception:
        logger.exception('Error adding secondary owner.')
```

4. Geben Sie den folgenden Befehl ein, um das Programm auszuführen.

```
python add_secondary_owner.py
```

Informationen zum Überprüfen dieser Änderungen im `vehicle-registration`-Ledger finden Sie unter Schritt 6: Den Revisionsverlauf für ein Dokument anzeigen.

Schritt 6: Den Revisionsverlauf für ein Dokument anzeigen

Nach dem Bearbeiten der Zulassungsdaten für ein Fahrzeug im vorherigen Schritt können Sie den Verlauf aller registrierten Eigentümer und alle anderen aktualisierten Felder abfragen. In diesem Schritt führen Sie eine Abfrage des Revisionsverlaufs eines Dokuments in der `VehicleRegistration`-Tabelle in Ihrem `vehicle-registration`-Ledger durch.

So zeigen Sie den Revisionsverlauf an

1. Verwenden Sie das folgende Programm (`query_history.py`), um den Versionsverlauf des `VehicleRegistration`-Dokuments mit VIN `1N4AL11D75C109151` abzufragen.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
```

```
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from datetime import datetime, timedelta
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import print_result, get_document_ids,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def format_date_time(date_time):
    """
    Format the given date time to a string.

    :type date_time: :py:class:`datetime.datetime`
    :param date_time: The date time to format.

    :rtype: str
    :return: The formatted date time.
    """
    return date_time.strftime('`%Y-%m-%dT%H:%M:%S.%fZ`')


def previous_primary_owners(driver, vin):
    """
    Find previous primary owners for the given VIN in a single transaction.
    In this example, query the `VehicleRegistration` history table to find all
 previous primary owners for a VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: VIN to find previous primary owners for.
    """
    person_ids = driver.execute_lambda(lambda executor:
 get_document_ids(executor,
```

```
 Constants.VEHICLE_REGISTRATION_TABLE_NAME,
                                                                        'VIN',
 vin))

    todays_date = datetime.utcnow() - timedelta(seconds=1)
    three_months_ago = todays_date - timedelta(days=90)
    query = 'SELECT data.Owners.PrimaryOwner, metadata.version FROM history({},
 {}, {}) AS h WHERE h.metadata.id = ?'.\
        format(Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 format_date_time(three_months_ago),
               format_date_time(todays_date))

    for ids in person_ids:
        logger.info("Querying the 'VehicleRegistration' table's history using
 VIN: {}.".format(vin))
        cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(query, ids))
        if not (print_result(cursor)) > 0:
            logger.info('No modification history found within the given time
 frame for document ID: {}'.format(ids))


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Query a table's history for a particular set of documents.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            vin = SampleData.VEHICLE_REGISTRATION[0]['VIN']
            previous_primary_owners(driver, vin)
            logger.info('Successfully queried history.')
    except Exception as e:
        logger.exception('Unable to query history to find previous owners.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from datetime import datetime, timedelta
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import print_result, get_document_ids,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def format_date_time(date_time):
    """
    Format the given date time to a string.

    :type date_time: :py:class:`datetime.datetime`
    :param date_time: The date time to format.

    :rtype: str
```

```
    :return: The formatted date time.
    """
    return date_time.strftime('`%Y-%m-%dT%H:%M:%S.%fZ`')


def previous_primary_owners(transaction_executor, vin):
    """
    Find previous primary owners for the given VIN in a single transaction.
    In this example, query the `VehicleRegistration` history table to find all
 previous primary owners for a VIN.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type vin: str
    :param vin: VIN to find previous primary owners for.
    """
    person_ids = get_document_ids(transaction_executor,
 Constants.VEHICLE_REGISTRATION_TABLE_NAME, 'VIN', vin)

    todays_date = datetime.utcnow() - timedelta(seconds=1)
    three_months_ago = todays_date - timedelta(days=90)
    query = 'SELECT data.Owners.PrimaryOwner, metadata.version FROM history({},
 {}, {}) AS h WHERE h.metadata.id = ?'.\
        format(Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 format_date_time(three_months_ago),
               format_date_time(todays_date))

    for ids in person_ids:
        logger.info("Querying the 'VehicleRegistration' table's history using
 VIN: {}.".format(vin))
        cursor = transaction_executor.execute_statement(query, ids)
        if not (print_result(cursor)) > 0:
            logger.info('No modification history found within the given time
 frame for document ID: {}'.format(ids))


if __name__ == '__main__':
    """
    Query a table's history for a particular set of documents.
    """
    try:
        with create_qldb_session() as session:
```

```
            vin = SampleData.VEHICLE_REGISTRATION[0]['VIN']
            session.execute_lambda(lambda lambda_executor:
previous_primary_owners(lambda_executor, vin),
                                   lambda retry_attempt: logger.info('Retrying
due to OCC conflict...'))
            logger.info('Successfully queried history.')
    except Exception:
        logger.exception('Unable to query history to find previous owners.')
```

> **Note**
>
> - Sie können den Revisionsverlauf eines Dokuments anzeigen, indem Sie die integrierte Verlaufsfunktion in der folgenden Syntax abfragen.
>
>   ```
>   SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
>   [ WHERE h.metadata.id = 'id' ]
>   ```
>
> - Die Start - und Endzeit sind beide optional. Es handelt sich um Amazon Ion-Literalwerte, die mit Backticks (`` `...` ``) gekennzeichnet werden können. Weitere Informationen hierzu finden Sie unter Ion mit PartiQL in Amazon QLDB abfragen.
> - Es hat sich bewährt, eine Verlaufsabfrage sowohl mit einem Datumsbereich (Start- und Endzeit) als auch mit einer Dokument-ID (`metadata.id`) zu qualifizieren. QLDB verarbeitet`SELECT` Abfragen in Transaktionen, für die ein Transaktions-Timeout-Limit gilt.
>
>   Der QLDB-Verlauf wird anhand der Dokument-ID indexiert, und Sie können derzeit keine zusätzlichen Verlaufsindizes erstellen. Verlaufsabfragen, die eine Start- und Endzeit enthalten, profitieren von der Qualifizierung des Datumsbereichs.

2. Geben Sie den folgenden Befehl ein, um das Programm auszuführen.

```
python query_history.py
```

Fahren Sie zum kryptografischen Überprüfen einer Dokumentrevision im `vehicle-registration`-Ledger mit Schritt 7: Verifizieren Sie ein Dokument in einem Ledger fort.

Schritt 7: Verifizieren Sie ein Dokument in einem Ledger

Mit Amazon QLDB können Sie die Integrität eines Dokuments im Journal Ihres Hauptbuchs effizient überprüfen, indem Sie kryptografisches Hashing mit SHA-256 verwenden. Weitere Informationen zur Funktionsweise von Verifizierung und kryptografischem Hashing in QLDB finden Sie unterDatenverifizierung in Amazon QLDB.

In diesem Schritt überprüfen Sie eine Dokumentrevision in der Tabelle `VehicleRegistration` in Ihrem `vehicle-registration`-Ledger. Zuerst fordern Sie einen Digest an, der als Ausgabedatei zurückgegeben wird und als Signatur des gesamten Änderungsverlaufs Ihres Ledgers fungiert. Anschließend fordern Sie einen Nachweis für die Revision in Bezug auf diesen Digest an. Mit diesem Nachweis wird die Integrität Ihrer Revision verifiziert, wenn alle Validierungsprüfungen bestanden werden.

So überprüfen Sie eine Dokumentrevision

1. Sehen Sie sich die folgenden `.py` Dateien an, die QLDB-Objekte darstellen, die für die Überprüfung erforderlich sind, und ein Hilfsmodul mit Hilfsfunktionen zur Konvertierung von QLDB-Antworttypen in Zeichenketten.

   1. `block_address.py`

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
```

```
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.


def block_address_to_dictionary(ion_dict):
    """
    Convert a block address from IonPyDict into a dictionary.
    Shape of the dictionary must be: {'IonText': "{strandId: <"strandId">,
 sequenceNo: <sequenceNo>}"}

    :type ion_dict: :py:class:`amazon.ion.simple_types.IonPyDict`/str
    :param ion_dict: The block address value to convert.

    :rtype: dict
    :return: The converted dict.
    """
    block_address = {'IonText': {}}
    if not isinstance(ion_dict, str):
        py_dict = '{{strandId: "{}", sequenceNo:
{}}}'.format(ion_dict['strandId'], ion_dict['sequenceNo'])
        ion_dict = py_dict
    block_address['IonText'] = ion_dict
    return block_address
```

2. verifier.py

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from array import array
from base64 import b64encode
from functools import reduce
from hashlib import sha256
from random import randrange

from amazon.ion.simpleion import loads

HASH_LENGTH = 32
UPPER_BOUND = 8


def parse_proof(value_holder):
    """
    Parse the Proof object returned by QLDB into an iterator.

    The Proof object returned by QLDB is a dictionary like the following:
    {'IonText': '[{{<hash>}},{{<hash>}}]'}

    :type value_holder: dict
    :param value_holder: A structure containing an Ion string value.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyList`
    :return: A list of hash values.
    """
    value_holder = value_holder.get('IonText')
    proof_list = loads(value_holder)
    return proof_list


def parse_block(value_holder):
    """
    Parse the Block object returned by QLDB and retrieve block hash.

    :type value_holder: dict
    :param value_holder: A structure containing an Ion string value.
```

```
    :rtype: :py:class:`amazon.ion.simple_types.IonPyBytes`
    :return: The block hash.
    """
    value_holder = value_holder.get('IonText')
    block = loads(value_holder)
    block_hash = block.get('blockHash')
    return block_hash


def flip_random_bit(original):
    """
    Flip a single random bit in the given hash value.
    This method is used to demonstrate QLDB's verification features.

    :type original: bytes
    :param original: The hash value to alter.

    :rtype: bytes
    :return: The altered hash with a single random bit changed.
    """
    assert len(original) != 0, 'Invalid bytes.'

    altered_position = randrange(len(original))
    bit_shift = randrange(UPPER_BOUND)
    altered_hash = bytearray(original).copy()

    altered_hash[altered_position] = altered_hash[altered_position] ^ (1 <<
 bit_shift)
    return bytes(altered_hash)


def compare_hash_values(hash1, hash2):
    """
    Compare two hash values by converting them into byte arrays, assuming they
 are little endian.

    :type hash1: bytes
    :param hash1: The hash value to compare.

    :type hash2: bytes
    :param hash2: The hash value to compare.

    :rtype: int
```

```
    :return: Zero if the hash values are equal, otherwise return the difference
 of the first pair of non-matching bytes.
    """
    assert len(hash1) == HASH_LENGTH
    assert len(hash2) == HASH_LENGTH

    hash_array1 = array('b', hash1)
    hash_array2 = array('b', hash2)

    for i in range(len(hash_array1) - 1, -1, -1):
        difference = hash_array1[i] - hash_array2[i]
        if difference != 0:
            return difference
    return 0


def join_hash_pairwise(hash1, hash2):
    """
    Take two hash values, sort them, concatenate them, and generate a new hash
 value from the concatenated values.

    :type hash1: bytes
    :param hash1: Hash value to concatenate.

    :type hash2: bytes
    :param hash2: Hash value to concatenate.

    :rtype: bytes
    :return: The new hash value generated from concatenated hash values.
    """
    if len(hash1) == 0:
        return hash2
    if len(hash2) == 0:
        return hash1

    concatenated = hash1 + hash2 if compare_hash_values(hash1, hash2) < 0 else
 hash2 + hash1
    new_hash_lib = sha256()
    new_hash_lib.update(concatenated)
    new_digest = new_hash_lib.digest()
    return new_digest


def calculate_root_hash_from_internal_hashes(internal_hashes, leaf_hash):
```

```
    """
    Combine the internal hashes and the leaf hash until only one root hash
 remains.

    :type internal_hashes: map
    :param internal_hashes: An iterable over a list of hash values.

    :type leaf_hash: bytes
    :param leaf_hash: The revision hash to pair with the first hash in the Proof
 hashes list.

    :rtype: bytes
    :return: The root hash constructed by combining internal hashes.
    """
    root_hash = reduce(join_hash_pairwise, internal_hashes, leaf_hash)
    return root_hash


def build_candidate_digest(proof, leaf_hash):
    """
    Build the candidate digest representing the entire ledger from the Proof
 hashes.

    :type proof: dict
    :param proof: The Proof object.

    :type leaf_hash: bytes
    :param leaf_hash: The revision hash to pair with the first hash in the Proof
 hashes list.

    :rtype: bytes
    :return: The calculated root hash.
    """
    parsed_proof = parse_proof(proof)
    root_hash = calculate_root_hash_from_internal_hashes(parsed_proof, leaf_hash)
    return root_hash


def verify_document(document_hash, digest, proof):
    """
    Verify document revision against the provided digest.

    :type document_hash: bytes
```

```
    :param document_hash: The SHA-256 value representing the document revision to
 be verified.

    :type digest: bytes
    :param digest: The SHA-256 hash value representing the ledger digest.

    :type proof: dict
    :param proof: The Proof object retrieved
 from :func:`pyqldbsamples.get_revision.get_revision`.

    :rtype: bool
    :return: If the document revision verify against the ledger digest.
    """
    candidate_digest = build_candidate_digest(proof, document_hash)
    return digest == candidate_digest


def to_base_64(input):
    """
    Encode input in base64.

    :type input: bytes
    :param input: Input to be encoded.

    :rtype: string
    :return: Return input that has been encoded in base64.
    """
    encoded_value = b64encode(input)
    return str(encoded_value, 'UTF-8')
```

3. qldb_string_utils.py

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
```

```
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
from amazon.ion.simpleion import dumps, loads


def value_holder_to_string(value_holder):
    """
    Returns the string representation of a given `value_holder`.

    :type value_holder: dict
    :param value_holder: The `value_holder` to convert to string.

    :rtype: str
    :return: The string representation of the supplied `value_holder`.
    """
    ret_val = dumps(loads(value_holder), binary=False, indent='  ',
 omit_version_marker=True)
    val = '{{ IonText: {}}}'.format(ret_val)
    return val


def block_response_to_string(block_response):
    """
    Returns the string representation of a given `block_response`.

    :type block_response: dict
    :param block_response: The `block_response` to convert to string.

    :rtype: str
    :return: The string representation of the supplied `block_response`.
    """
    string = ''
    if block_response.get('Block', {}).get('IonText') is not None:
        string += 'Block: ' + value_holder_to_string(block_response['Block']
['IonText']) + ', '
```

```
    if block_response.get('Proof', {}).get('IonText') is not None:
        string += 'Proof: ' + value_holder_to_string(block_response['Proof']
['IonText'])

    return '{' + string + '}'


def digest_response_to_string(digest_response):
    """
    Returns the string representation of a given `digest_response`.

    :type digest_response: dict
    :param digest_response: The `digest_response` to convert to string.

    :rtype: str
    :return: The string representation of the supplied `digest_response`.
    """
    string = ''
    if digest_response.get('Digest') is not None:
        string += 'Digest: ' + str(digest_response['Digest']) + ', '

    if digest_response.get('DigestTipAddress', {}).get('IonText') is not None:
        string += 'DigestTipAddress: ' +
 value_holder_to_string(digest_response['DigestTipAddress']['IonText'])

    return '{' + string + '}'
```

2. Verwenden Sie zwei `.py`-Programme (`get_digest.py` und `get_revision.py`), um die folgenden Schritte auszuführen:

   - Fordern Sie einen neuen Digest aus dem Ledger `vehicle-registration` an.
   - Fordern Sie einen Nachweis für jede Revision des Dokuments mit VIN `1N4AL11D75C109151` aus der `VehicleRegistration`-Tabelle an.
   - Überprüfen Sie die Revisionen mit dem zurückgegebenen Digest und dem Nachweis, indem Sie den Digest neu berechnen.

   Das Programm `get_digest.py` enthält den folgenden Code.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
```

```
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from boto3 import client

from pyqldbsamples.constants import Constants
from pyqldbsamples.qldb.qldb_string_utils import digest_response_to_string

logger = getLogger(__name__)
basicConfig(level=INFO)
qldb_client = client('qldb')


def get_digest_result(name):
    """
    Get the digest of a ledger's journal.

    :type name: str
    :param name: Name of the ledger to operate on.

    :rtype: dict
    :return: The digest in a 256-bit hash value and a block address.
    """
    logger.info("Let's get the current digest of the ledger named {}".format(name))
```

```
    result = qldb_client.get_digest(Name=name)
    logger.info('Success. LedgerDigest:
 {}.'.format(digest_response_to_string(result)))
    return result


def main(ledger_name=Constants.LEDGER_NAME):
    """
    This is an example for retrieving the digest of a particular ledger.
    """
    try:
        get_digest_result(ledger_name)
    except Exception as e:
        logger.exception('Unable to get a ledger digest!')
        raise e


if __name__ == '__main__':
    main()
```

> **Note**
>
> Verwenden Sie die `get_digest_result`-Funktion, um einen Digest anzufordern, der die aktuelle Spitze des Journals in Ihrem Ledger abdeckt. Die Notiz des Journals bezieht sich auf den letzten übernommenen Block zu dem Zeitpunkt, zu dem QLDB Ihre Anfrage erhält.

Das Programm `get_revision.py` enthält den folgenden Code.

3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
```

```
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from amazon.ion.simpleion import loads
from boto3 import client

from pyqldbsamples.constants import Constants
from pyqldbsamples.get_digest import get_digest_result
from pyqldbsamples.model.sample_data import SampleData, convert_object_to_ion
from pyqldbsamples.qldb.block_address import block_address_to_dictionary
from pyqldbsamples.verifier import verify_document, flip_random_bit, to_base_64
from pyqldbsamples.connect_to_ledger import create_qldb_driver
from pyqldbsamples.qldb.qldb_string_utils import value_holder_to_string

logger = getLogger(__name__)
basicConfig(level=INFO)
qldb_client = client('qldb')


def get_revision(ledger_name, document_id, block_address, digest_tip_address):
    """
    Get the revision data object for a specified document ID and block address.
    Also returns a proof of the specified revision for verification.

    :type ledger_name: str
    :param ledger_name: Name of the ledger containing the document to query.

    :type document_id: str
```

```
    :param document_id: Unique ID for the document to be verified, contained in
 the committed view of the document.

    :type block_address: dict
    :param block_address: The location of the block to request.

    :type digest_tip_address: dict
    :param digest_tip_address: The latest block location covered by the digest.

    :rtype: dict
    :return: The response of the request.
    """
    result = qldb_client.get_revision(Name=ledger_name,
 BlockAddress=block_address, DocumentId=document_id,
                                      DigestTipAddress=digest_tip_address)
    return result


def lookup_registration_for_vin(driver, vin):
    """
    Query revision history for a particular vehicle for verification.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: VIN to query the revision history of a specific registration
 with.

    :rtype: :py:class:`pyqldb.cursor.buffered_cursor.BufferedCursor`
    :return: Cursor on the result set of the statement query.
    """
    logger.info("Querying the 'VehicleRegistration' table for VIN:
 {}...".format(vin))
    query = 'SELECT * FROM _ql_committed_VehicleRegistration WHERE data.VIN = ?'
    return driver.execute_lambda(lambda txn: txn.execute_statement(query,
 convert_object_to_ion(vin)))


def verify_registration(driver, ledger_name, vin):
    """
    Verify each version of the registration for the given VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
```

```
    :param driver: An instance of the QldbDriver class.

    :type ledger_name: str
    :param ledger_name: The ledger to get digest from.

    :type vin: str
    :param vin: VIN to query the revision history of a specific registration
with.

    :raises AssertionError: When verification failed.
    """
    logger.info("Let's verify the registration with VIN = {}, in ledger =
{}.".format(vin, ledger_name))
    digest = get_digest_result(ledger_name)
    digest_bytes = digest.get('Digest')
    digest_tip_address = digest.get('DigestTipAddress')

    logger.info('Got a ledger digest: digest tip address = {}, digest =
{}.'.format(
        value_holder_to_string(digest_tip_address.get('IonText')),
to_base_64(digest_bytes)))

    logger.info('Querying the registration with VIN = {} to verify each version
of the registration...'.format(vin))
    cursor = lookup_registration_for_vin(driver, vin)
    logger.info('Getting a proof for the document.')

    for row in cursor:
        block_address = row.get('blockAddress')
        document_id = row.get('metadata').get('id')

        result = get_revision(ledger_name, document_id,
block_address_to_dictionary(block_address), digest_tip_address)
        revision = result.get('Revision').get('IonText')
        document_hash = loads(revision).get('hash')

        proof = result.get('Proof')
        logger.info('Got back a proof: {}.'.format(proof))

        verified = verify_document(document_hash, digest_bytes, proof)
        if not verified:
            raise AssertionError('Document revision is not verified.')
        else:
            logger.info('Success! The document is verified.')
```

```
        altered_document_hash = flip_random_bit(document_hash)
        logger.info("Flipping one bit in the document's hash and assert that the
 document is NOT verified. "
                    "The altered document hash is:
 {}.".format(to_base_64(altered_document_hash)))
        verified = verify_document(altered_document_hash, digest_bytes, proof)
        if verified:
            raise AssertionError('Expected altered document hash to not be
 verified against digest.')
        else:
            logger.info('Success! As expected flipping a bit in the document
 hash causes verification to fail.')

        logger.info('Finished verifying the registration with VIN = {} in ledger
 = {}.'.format(vin, ledger_name))


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Verify the integrity of a document revision in a QLDB ledger.
    """
    registration = SampleData.VEHICLE_REGISTRATION[0]
    vin = registration['VIN']
    try:
        with create_qldb_driver(ledger_name) as driver:
            verify_registration(driver, ledger_name, vin)
    except Exception as e:
        logger.exception('Unable to verify revision.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
```

```
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from amazon.ion.simpleion import loads
from boto3 import client

from pyqldbsamples.constants import Constants
from pyqldbsamples.get_digest import get_digest_result
from pyqldbsamples.model.sample_data import SampleData, convert_object_to_ion
from pyqldbsamples.qldb.block_address import block_address_to_dictionary
from pyqldbsamples.verifier import verify_document, flip_random_bit, to_base_64
from pyqldbsamples.connect_to_ledger import create_qldb_session
from pyqldbsamples.qldb.qldb_string_utils import value_holder_to_string

logger = getLogger(__name__)
basicConfig(level=INFO)
qldb_client = client('qldb')


def get_revision(ledger_name, document_id, block_address, digest_tip_address):
    """
    Get the revision data object for a specified document ID and block address.
    Also returns a proof of the specified revision for verification.

    :type ledger_name: str
```

```
    :param ledger_name: Name of the ledger containing the document to query.

    :type document_id: str
    :param document_id: Unique ID for the document to be verified, contained in
 the committed view of the document.

    :type block_address: dict
    :param block_address: The location of the block to request.

    :type digest_tip_address: dict
    :param digest_tip_address: The latest block location covered by the digest.

    :rtype: dict
    :return: The response of the request.
    """
    result = qldb_client.get_revision(Name=ledger_name,
 BlockAddress=block_address, DocumentId=document_id,
                                      DigestTipAddress=digest_tip_address)
    return result


def lookup_registration_for_vin(qldb_session, vin):
    """
    Query revision history for a particular vehicle for verification.

    :type qldb_session: :py:class:`pyqldb.session.qldb_session.QldbSession`
    :param qldb_session: An instance of the QldbSession class.

    :type vin: str
    :param vin: VIN to query the revision history of a specific registration
 with.

    :rtype: :py:class:`pyqldb.cursor.buffered_cursor.BufferedCursor`
    :return: Cursor on the result set of the statement query.
    """
    logger.info("Querying the 'VehicleRegistration' table for VIN:
 {}...".format(vin))
    query = 'SELECT * FROM _ql_committed_VehicleRegistration WHERE data.VIN = ?'
    parameters = [convert_object_to_ion(vin)]
    cursor = qldb_session.execute_statement(query, parameters)
    return cursor


def verify_registration(qldb_session, ledger_name, vin):
```

```
    """
    Verify each version of the registration for the given VIN.

    :type qldb_session: :py:class:`pyqldb.session.qldb_session.QldbSession`
    :param qldb_session: An instance of the QldbSession class.

    :type ledger_name: str
    :param ledger_name: The ledger to get digest from.

    :type vin: str
    :param vin: VIN to query the revision history of a specific registration
with.

    :raises AssertionError: When verification failed.
    """
    logger.info("Let's verify the registration with VIN = {}, in ledger =
{}.".format(vin, ledger_name))
    digest = get_digest_result(ledger_name)
    digest_bytes = digest.get('Digest')
    digest_tip_address = digest.get('DigestTipAddress')

    logger.info('Got a ledger digest: digest tip address = {}, digest =
{}.'.format(
        value_holder_to_string(digest_tip_address.get('IonText')),
to_base_64(digest_bytes)))

    logger.info('Querying the registration with VIN = {} to verify each version
of the registration...'.format(vin))
    cursor = lookup_registration_for_vin(qldb_session, vin)
    logger.info('Getting a proof for the document.')

    for row in cursor:
        block_address = row.get('blockAddress')
        document_id = row.get('metadata').get('id')

        result = get_revision(ledger_name, document_id,
block_address_to_dictionary(block_address), digest_tip_address)
        revision = result.get('Revision').get('IonText')
        document_hash = loads(revision).get('hash')

        proof = result.get('Proof')
        logger.info('Got back a proof: {}.'.format(proof))

        verified = verify_document(document_hash, digest_bytes, proof)
```

```
        if not verified:
            raise AssertionError('Document revision is not verified.')
        else:
            logger.info('Success! The document is verified.')

        altered_document_hash = flip_random_bit(document_hash)
        logger.info("Flipping one bit in the document's hash and assert that the
 document is NOT verified. "
                    "The altered document hash is:
 {}.".format(to_base_64(altered_document_hash)))
        verified = verify_document(altered_document_hash, digest_bytes, proof)
        if verified:
            raise AssertionError('Expected altered document hash to not be
 verified against digest.')
        else:
            logger.info('Success! As expected flipping a bit in the document
 hash causes verification to fail.')

        logger.info('Finished verifying the registration with VIN = {} in ledger
 = {}.'.format(vin, ledger_name))


if __name__ == '__main__':
    """
    Verify the integrity of a document revision in a QLDB ledger.
    """
    registration = SampleData.VEHICLE_REGISTRATION[0]
    vin = registration['VIN']
    try:
        with create_qldb_session() as session:
            verify_registration(session, Constants.LEDGER_NAME, vin)
    except Exception:
        logger.exception('Unable to verify revision.')
```

> **Note**
>
> Nachdem die `get_revision`-Funktion einen Nachweis für die angegebene Dokumentrevision zurückgibt, verwendet dieses Programm eine clientseitige API, um diese Revision zu überprüfen.

3. Geben Sie den folgenden Befehl ein, um das Programm auszuführen.

```
python get_revision.py
```

Wenn Sie den `vehicle-registration`-Ledger nicht mehr verwenden müssen, fahren Sie mit Schritt 8 (optional): Bereinigen von Ressourcen fort.

Schritt 8 (optional): Bereinigen von Ressourcen

Sie können den `vehicle-registration`-Ledger weiterhin verwenden. Wenn Sie ihn nicht mehr benötigen, sollten Sie ihn jedoch löschen.

So löschen Sie den Ledger

1. Verwenden Sie das folgende Programm (`delete_ledger.py`), um Ihren `vehicle-registration`-Ledger und all seine Inhalte zu löschen.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO
from time import sleep
```

```
from boto3 import client

from pyqldbsamples.constants import Constants
from pyqldbsamples.describe_ledger import describe_ledger

logger = getLogger(__name__)
basicConfig(level=INFO)
qldb_client = client('qldb')

LEDGER_DELETION_POLL_PERIOD_SEC = 20


def delete_ledger(ledger_name):
    """
    Send a request to QLDB to delete the specified ledger.

    :type ledger_name: str
    :param ledger_name: Name for the ledger to be deleted.

    :rtype: dict
    :return: Result from the request.
    """
    logger.info('Attempting to delete the ledger with name:
 {}...'.format(ledger_name))
    result = qldb_client.delete_ledger(Name=ledger_name)
    logger.info('Success.')
    return result


def wait_for_deleted(ledger_name):
    """
    Wait for the ledger to be deleted.

    :type ledger_name: str
    :param ledger_name: The ledger to check on.
    """
    logger.info('Waiting for the ledger to be deleted...')
    while True:
        try:
            describe_ledger(ledger_name)
            logger.info('The ledger is still being deleted. Please wait...')
            sleep(LEDGER_DELETION_POLL_PERIOD_SEC)
        except qldb_client.exceptions.ResourceNotFoundException:
```

```
            logger.info('Success. The ledger is deleted.')
            break


def set_deletion_protection(ledger_name, deletion_protection):
    """
    Update an existing ledger's deletion protection.

    :type ledger_name: str
    :param ledger_name: Name of the ledger to update.

    :type deletion_protection: bool
    :param deletion_protection: Enable or disable the deletion protection.

    :rtype: dict
    :return: Result from the request.
    """
    logger.info("Let's set deletion protection to {} for the ledger with name
 {}.".format(deletion_protection,

       ledger_name))
    result = qldb_client.update_ledger(Name=ledger_name,
 DeletionProtection=deletion_protection)
    logger.info('Success. Ledger updated: {}'.format(result))


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Delete a ledger.
    """
    try:
        set_deletion_protection(ledger_name, False)
        delete_ledger(ledger_name)
        wait_for_deleted(ledger_name)
    except Exception as e:
        logger.exception('Unable to delete the ledger.')
        raise e


if __name__ == '__main__':
    main()
```

> **Note**
>
> Wenn der Löschschutz für Ihr Ledger aktiviert ist, müssen Sie ihn zunächst deaktivieren, bevor Sie das Ledger mithilfe der QLDB-API löschen können.

Die Datei `delete_ledger.py` weist auch eine Abhängigkeit vom folgenden Programm (`describe_ledger.py`) auf.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from boto3 import client

from pyqldbsamples.constants import Constants

logger = getLogger(__name__)
basicConfig(level=INFO)
```

```
qldb_client = client('qldb')


def describe_ledger(ledger_name):
    """
    Describe a ledger.

    :type ledger_name: str
    :param ledger_name: Name of the ledger to describe.
    """
    logger.info('describe ledger with name: {}.'.format(ledger_name))
    result = qldb_client.describe_ledger(Name=ledger_name)
    result.pop('ResponseMetadata')
    logger.info('Success. Ledger description: {}.'.format(result))
    return result


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Describe a QLDB ledger.
    """
    try:
        describe_ledger(ledger_name)
    except Exception as e:
        logger.exception('Unable to describe a ledger.')
        raise e


if __name__ == '__main__':
    main()
```

2. Geben Sie den folgenden Befehl ein, um das Programm auszuführen.

```
python delete_ledger.py
```

Arbeiten mit Amazon Ion-Datentypen in Amazon QLDB

Amazon QLDB speichert seine Daten im Amazon Ion-Format. Um mit Daten in QLDB arbeiten zu können, müssen Sie eine Ion-Bibliothek als Abhängigkeit für eine unterstützte Programmiersprache verwenden.

In diesem Abschnitt erfahren Sie, wie Sie Daten von nativen Typen in ihre Ionenäquivalente konvertieren und umgekehrt. Dieses Referenzhandbuch zeigt Codebeispiele, die den QLDB-Treiber verwenden, um Ion-Daten in einem QLDB-Ledger zu verarbeiten. Es enthält Codebeispiele für Java, .NETTypeScript (Node.js

Themen

Voraussetzungen

In den folgenden Codebeispielen wird davon ausgegangen, dass Sie über eine QLDB-Treiberinstanz verfügen, die mit einem aktiven Ledger mit einer Tabelle namens verbunden ist`ExampleTable`. Die Tabelle enthält ein einzelnes vorhandenes Dokument mit den folgenden acht Feldern:

- ExampleBool
- ExampleInt
- ExampleFloat
- ExampleDecimal
- ExampleTimestamp
- ExampleString
- ExampleBlob
- ExampleList

> **Note**
>
> Gehen Sie für diese Referenz davon aus, dass der in jedem Feld gespeicherte Typ mit seinem Namen übereinstimmt. In der Praxis erzwingt QLDB keine Schema- oder Datentypdefinitionen für Dokumentfelder.

Bool

Die folgenden Codebeispiele zeigen, wie der Boole-Typ Ion verarbeitet werden.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

boolean exampleBoolean = driver.execute((txn) -> {
    // Transforming a Java boolean to Ion
    boolean aBoolean = true;
    IonValue ionBool = ionSystem.newBool(aBoolean);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleBool = ?", ionBool);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleBool from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java boolean
        // Cast IonValue to IonBool first
        aBoolean = ((IonBool)ionValue).booleanValue();
    }

    // exampleBoolean is now the value fetched from QLDB
    return aBoolean;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...
```

```
IValueFactory valueFactory = new ValueFactory();

// Transforming a C# bool to Ion.
bool nativeBool  = true;
IIonValue ionBool = valueFactory.NewBool(nativeBool);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleBool = ?", ionBool);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleBool from ExampleTable");
});

bool? retrievedBool = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# bool.
    retrievedBool = ionValue.BoolValue;
}
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

Go

```
exampleBool, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aBool := true

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleBool = ?", aBool)
 if err != nil {
  return nil, err
 }
```

```
 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleBool FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult bool
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleBool is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonBoolean(driver: QldbDriver): Promise<boolean> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleBool = ?", true);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleBool FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to a TypeScript Boolean
            const boolValue: boolean = ionValue.booleanValue();
            return boolValue;
        })
    );
}
```

Python

```
def update_and_query_ion_bool(txn):
    # QLDB can take in a Python bool
    a_bool = True

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleBool = ?", a_bool)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleBool FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python bool is a child class of Python bool
        a_bool = ion_value

    # example_bool is now the value fetched from QLDB
    return a_bool


example_bool = driver.execute_lambda(lambda txn: update_and_query_ion_bool(txn))
```

Int

Die folgenden Codebeispiele zeigen, wie der Ion-Integer-Typ verarbeitet werden.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

int exampleInt = driver.execute((txn) -> {
    // Transforming a Java int to Ion
    int aInt = 256;
    IonValue ionInt = ionSystem.newInt(aInt);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleInt = ?", ionInt);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleInt from ExampleTable");
```

```
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java int
        // Cast IonValue to IonInt first
        aInt = ((IonInt)ionValue).intValue();
    }

    // exampleInt is now the value fetched from QLDB
    return aInt;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Transforming a C# int to Ion.
int nativeInt  = 256;
IIonValue ionInt = valueFactory.NewInt(nativeInt);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleInt = ?", ionInt);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleInt from ExampleTable");
});

int? retrievedInt = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# int.
    retrievedInt = ionValue.IntValue;
}
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

Go

```
exampleInt, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aInt := 256

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleInt = ?", aInt)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleInt FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult int
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleInt is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonInt(driver: QldbDriver): Promise<number> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
```

```
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleInt = ?", 256);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleInt FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to a TypeScript Number
            const intValue: number = ionValue.numberValue();
            return intValue;
        })
    );
}
```

Python

```
def update_and_query_ion_int(txn):
    # QLDB can take in a Python int
    a_int = 256

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleInt = ?", a_int)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleInt FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python int is a child class of Python int
        a_int = ion_value

    # example_int is now the value fetched from QLDB
    return a_int


example_int = driver.execute_lambda(lambda txn: update_and_query_ion_int(txn))
```

Gleitkommazahl

Die folgenden Codebeispiele zeigen, wie der Ion-Float-Typ verarbeitet werden.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

float exampleFloat = driver.execute((txn) -> {
    // Transforming a Java float to Ion
    float aFloat = 256;
    IonValue ionFloat = ionSystem.newFloat(aFloat);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleFloat = ?", ionFloat);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleFloat from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java float
        // Cast IonValue to IonFloat first
        aFloat = ((IonFloat)ionValue).floatValue();
    }

    // exampleFloat is now the value fetched from QLDB
    return aFloat;
});
```

.NET

Verwenden von C# float

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Transforming a C# float to Ion.
float nativeFloat = 256;
IIonValue ionFloat = valueFactory.NewFloat(nativeFloat);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
```

```
    await txn.Execute("UPDATE ExampleTable SET ExampleFloat = ?", ionFloat);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleFloat from ExampleTable");
});

float? retrievedFloat = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# float. We cast ionValue.DoubleValue to a float
    // but be cautious, this is a down-cast and can lose precision.
    retrievedFloat = (float)ionValue.DoubleValue;
}
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

C# double verwenden

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Transforming a C# double to Ion.
double nativeDouble = 256;
IIonValue ionFloat = valueFactory.NewFloat(nativeDouble);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleFloat = ?", ionFloat);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleFloat from ExampleTable");
});
```

```
double? retrievedDouble = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# double.
    retrievedDouble = ionValue.DoubleValue;
}
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

Go

```
exampleFloat, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aFloat := float32(256)

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleFloat = ?", aFloat)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleFloat FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  // float64 would work as well
  var decodedResult float32
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }
```

```
  // exampleFloat is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonFloat(driver: QldbDriver): Promise<number> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleFloat = ?", 25.6);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleFloat FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to a TypeScript Number
            const floatValue: number = ionValue.numberValue();
            return floatValue;
        })
    );
}
```

Python

```
def update_and_query_ion_float(txn):
    # QLDB can take in a Python float
    a_float = float(256)

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleFloat = ?", a_float)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleFloat FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python float is a child class of Python float
        a_float = ion_value
```

```
    # example_float is now the value fetched from QLDB
    return a_float


example_float = driver.execute_lambda(lambda txn: update_and_query_ion_float(txn))
```

Dezimal

Die folgenden Codebeispiele zeigen, wie der Ion-Dezimaltyp verarbeitet werden.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

double exampleDouble = driver.execute((txn) -> {
    // Transforming a Java double to Ion
    double aDouble = 256;
    IonValue ionDecimal = ionSystem.newDecimal(aDouble);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleDecimal = ?", ionDecimal);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleDecimal from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java double
        // Cast IonValue to IonDecimal first
        aDouble = ((IonDecimal)ionValue).doubleValue();
    }

    // exampleDouble is now the value fetched from QLDB
    return aDouble;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...
```

```
IValueFactory valueFactory = new ValueFactory();

// Transforming a C# decimal to Ion.
decimal nativeDecimal = 256.8723m;
IIonValue ionDecimal = valueFactory.NewDecimal(nativeDecimal);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleDecimal = ?", ionDecimal);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleDecimal from ExampleTable");
});

decimal? retrievedDecimal = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# decimal.
    retrievedDecimal = ionValue.DecimalValue;
}
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

Go

```
exampleDecimal, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aDecimal, err := ion.ParseDecimal("256")
 if err != nil {
  return nil, err
 }

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleDecimal = ?", aDecimal)
 if err != nil {
```

```
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleDecimal FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult ion.Decimal
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleDecimal is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonDecimal(driver: QldbDriver): Promise<Decimal> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            // Creating a Decimal value. Decimal is an Ion Type with high precision
            let ionDecimal: Decimal = dom.load("2.5d-6").decimalValue();
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleDecimal = ?",
 ionDecimal);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleDecimal FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Get the Ion Decimal
            ionDecimal = ionValue.decimalValue();
            return ionDecimal;
        })
```

```
    );
}
```

> **Note**
>
> Sie können es auch verwendenionValue.numberValue(), um eine Ionen-Dezimalzahl in eine JavaScript Zahl umzuwandeln. ionValue.numberValue()Die Verwendung hat eine bessere Leistung, ist aber weniger präzise.

Python

```
def update_and_query_ion_decimal(txn):
    # QLDB can take in a Python decimal
    a_decimal = Decimal(256)

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleDecimal = ?", a_decimal)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleDecimal FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python decimal is a child class of Python decimal
        a_decimal = ion_value

    # example_decimal is now the value fetched from QLDB
    return a_decimal


example_decimal = driver.execute_lambda(lambda txn:
 update_and_query_ion_decimal(txn))
```

Zeitstempel

Die folgenden Codebeispiele zeigen, wie der Ion-Zeitstempeltyp verarbeitet werden.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

Date exampleDate = driver.execute((txn) -> {
    // Transforming a Java Date to Ion
    Date aDate = new Date();
    IonValue ionTimestamp = ionSystem.newUtcTimestamp(aDate);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleTimestamp = ?", ionTimestamp);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleTimestamp from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java Date
        // Cast IonValue to IonTimestamp first
        aDate = ((IonTimestamp)ionValue).dateValue();
    }

    // exampleDate is now the value fetched from QLDB
    return aDate;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
using Amazon.IonDotnet;
...

IValueFactory valueFactory = new ValueFactory();

// Convert C# native DateTime to Ion.
DateTime nativeDateTime = DateTime.Now;

// First convert it to a timestamp object from Ion.
Timestamp timestamp = new Timestamp(nativeDateTime);
// Then convert to Ion timestamp.
IIonValue ionTimestamp = valueFactory.NewTimestamp(timestamp);
```

```
IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleTimestamp = ?", ionTimestamp);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleTimestamp from ExampleTable");
});

DateTime? retrievedDateTime = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# DateTime.
    retrievedDateTime = ionValue.TimestampValue.DateTimeValue;
}
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

Go

```
exampleTimestamp, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aTimestamp := time.Date(2006, time.May, 20, 12, 30, 0, 0, time.UTC)
 if err != nil {
  return nil, err
 }

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleTimestamp = ?", aTimestamp)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleTimestamp FROM ExampleTable")
 if err != nil {
```

```
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult ion.Timestamp
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleTimestamp is now the value fetched from QLDB
  return decodedResult.GetDateTime(), nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonTimestamp(driver: QldbDriver): Promise<Date> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            let exampleDateTime: Date = new Date(Date.now());
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleTimestamp = ?",
 exampleDateTime);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleTimestamp FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to a TypeScript Date
            exampleDateTime = ionValue.timestampValue().getDate();
            return exampleDateTime;
        })
    );
}
```

Python

```
def update_and_query_ion_timestamp(txn):
    # QLDB can take in a Python timestamp
```

```
    a_datetime = datetime.now()

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleTimestamp = ?",
 a_datetime)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleTimestamp FROM
 ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python timestamp is a child class of Python datetime
        a_timestamp = ion_value

    # example_timestamp is now the value fetched from QLDB
    return a_timestamp


example_timestamp = driver.execute_lambda(lambda txn:
 update_and_query_ion_timestamp(txn))
```

Zeichenfolge

Die folgenden Codebeispiele zeigen, wie der Ion-Stringtyp verarbeitet werden.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

String exampleString = driver.execute((txn) -> {
    // Transforming a Java String to Ion
    String aString = "Hello world!";
    IonValue ionString = ionSystem.newString(aString);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleString = ?", ionString);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleString from ExampleTable");
    // Assume there is only one document in ExampleTable
```

```
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java String
        // Cast IonValue to IonString first
        aString = ((IonString)ionValue).stringValue();
    }

    // exampleString is now the value fetched from QLDB
    return aString;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Convert C# string to Ion.
String nativeString = "Hello world!";
IIonValue ionString = valueFactory.NewString(nativeString);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleString = ?", ionString);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleString from ExampleTable");
});

String retrievedString = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# string.
    retrievedString = ionValue.StringValue;
}
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

Go

```
exampleString, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aString := "Hello World!"

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleString = ?", aString)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleString FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult string
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleString is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonString(driver: QldbDriver): Promise<string> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
```

```
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleString = ?", "Hello
 World!");

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleString FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to a TypeScript String
            const stringValue: string = ionValue.stringValue();
            return stringValue;
        })
    );
}
```

Python

```
def update_and_query_ion_string(txn):
    # QLDB can take in a Python string
    a_string = "Hello world!"

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleString = ?", a_string)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleString FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python string is a child class of Python string
        a_string = ion_value

    # example_string is now the value fetched from QLDB
    return a_string


example_string = driver.execute_lambda(lambda txn: update_and_query_ion_string(txn))
```

Blob

Die folgenden Codebeispiele zeigen, wie der Ion-Blob-Typ verarbeitet werden.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

byte[] exampleBytes = driver.execute((txn) -> {
    // Transforming a Java byte array to Ion
    // Transform any arbitrary data to a byte array to store in QLDB
    String aString = "Hello world!";
    byte[] aByteArray = aString.getBytes();
    IonValue ionBlob = ionSystem.newBlob(aByteArray);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleBlob = ?", ionBlob);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleBlob from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java byte array
        // Cast IonValue to IonBlob first
        aByteArray = ((IonBlob)ionValue).getBytes();
    }

    // exampleBytes is now the value fetched from QLDB
    return aByteArray;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
using System.Text;
...

IValueFactory valueFactory = new ValueFactory();

// Transform any arbitrary data to a byte array to store in QLDB.
string nativeString = "Hello world!";
```

```
byte[] nativeByteArray = Encoding.UTF8.GetBytes(nativeString);
// Transforming a C# byte array to Ion.
IIonValue ionBlob = valueFactory.NewBlob(nativeByteArray);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleBlob = ?", ionBlob);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleBlob from ExampleTable");
});

byte[] retrievedByteArray = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# byte array.
    retrievedByteArray = ionValue.Bytes().ToArray();
}
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

Go

```
exampleBlob, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aBlob := []byte("Hello World!")

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleBlob = ?", aBlob)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleBlob FROM ExampleTable")
```

```
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult []byte
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleBlob is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonBlob(driver: QldbDriver): Promise<Uint8Array> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            const enc = new TextEncoder();
            let blobValue: Uint8Array = enc.encode("Hello World!");
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleBlob = ?", blobValue);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleBlob FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to TypeScript Uint8Array
            blobValue = ionValue.uInt8ArrayValue();
            return blobValue;
        })
    );
}
```

Python

```
def update_and_query_ion_blob(txn):
```

```
    # QLDB can take in a Python byte array
    a_string = "Hello world!"
    a_byte_array = str.encode(a_string)

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleBlob = ?", a_byte_array)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleBlob FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python blob is a child class of Python byte array
        a_blob = ion_value

    # example_blob is now the value fetched from QLDB
    return a_blob


example_blob = driver.execute_lambda(lambda txn: update_and_query_ion_blob(txn))
```

Liste

Die folgenden Codebeispiele zeigen, wie Sie den Ion-List-Typen verarbeiten.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

List<Integer> exampleList = driver.execute((txn) -> {
    // Transforming a Java List to Ion
    List<Integer> aList = new ArrayList<>();
    // Add 5 Integers to the List for the sake of example
    for (int i = 0; i < 5; i++) {
        aList.add(i);
    }
    // Create an empty Ion List
    IonList ionList = ionSystem.newEmptyList();
    // Add the 5 Integers to the Ion List
    for (Integer i : aList) {
        // Convert each Integer to Ion ints first to add it to the Ion List
```

```
        ionList.add(ionSystem.newInt(i));
    }

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleList = ?", (IonValue) ionList);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleList from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Iterate through the Ion List to map it to a Java List
        List<Integer> intList = new ArrayList<>();
        for (IonValue ionInt : (IonList)ionValue) {
            // Convert the 5 Ion ints to Java Integers
            intList.add(((IonInt)ionInt).intValue());
        }
        aList = intList;
    }

    // exampleList is now the value fetched from QLDB
    return aList;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
using System.Collections.Generic;
...

IValueFactory valueFactory = new ValueFactory();

// Transforming a C# list to Ion.
IIonValue ionList = valueFactory.NewEmptyList();
foreach (int i in new List<int> {0, 1, 2, 3, 4})
{
    // Convert to Ion int and add to Ion list.
    ionList.Add(valueFactory.NewInt(i));
}

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
```

```
    await txn.Execute("UPDATE ExampleTable SET ExampleList = ?", ionList);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleList from ExampleTable");
});


List<int> retrievedList = new List<int>();
await foreach (IIonValue ionValue in selectResult)
{
    // Iterate through the Ion List to map it to a C# list.
    foreach (IIonValue ionInt in ionValue)
    {
        retrievedList.Add(ionInt.IntValue);
    }
}
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

Go

```
exampleList, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aList := []int{1, 2, 3, 4, 5}

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleList = ?", aList)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleList FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
```

```
 if result.Next(txn) {
  var decodedResult []int
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleList is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonList(driver: QldbDriver): Promise<number[]> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            let listOfNumbers: number[] = [1, 2, 3, 4, 5];
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleList = ?",
 listOfNumbers);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleList FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Get Ion List
            const ionList: dom.Value[] = ionValue.elements();
            // Iterate through the Ion List to map it to a JavaScript Array
            let intList: number[] = [];
            ionList.forEach(item => {
                // Transforming Ion to a TypeScript Number
                const intValue: number = item.numberValue();
                intList.push(intValue);
            });
            listOfNumbers = intList;
            return listOfNumbers;
        })
    );
}
```

Python

```
def update_and_query_ion_list(txn):
    # QLDB can take in a Python list
    a_list = list()
    for i in range(0, 5):
        a_list.append(i)

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleList = ?", a_list)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleList FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python blob is a child class of Python list
        a_list = ion_value

    # example_list is now the value fetched from QLDB
    return a_list


example_list = driver.execute_lambda(lambda txn: update_and_query_ion_list(txn))
```

Struct

In QLDB unterscheiden sich`struct` Datentypen besonders von anderen Ionentypen. Dokumente auf oberster Ebene, die Sie in eine Tabelle einfügen, müssen von diesem`struct` Typ sein. In einem Dokumentenfeld kann auch ein verschachteltes Feld gespeichert werden`struct`.

Der Einfachheit halber wird in den folgenden Beispielen ein Dokument definiert, das nur die`ExampleInt` Felder`ExampleString` und enthält.

Java

```
class ExampleStruct {
    public String exampleString;
    public int exampleInt;

    public ExampleStruct(String exampleString, int exampleInt) {
```

```
        this.exampleString = exampleString;
        this.exampleInt = exampleInt;
    }
}

// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

ExampleStruct examplePojo = driver.execute((txn) -> {
    // Transforming a POJO to Ion
    ExampleStruct aPojo = new ExampleStruct("Hello world!", 256);
    // Create an empty Ion struct
    IonStruct ionStruct = ionSystem.newEmptyStruct();
    // Map the fields of the POJO to Ion values and put them in the Ion struct
    ionStruct.add("ExampleString", ionSystem.newString(aPojo.exampleString));
    ionStruct.add("ExampleInt", ionSystem.newInt(aPojo.exampleInt));

    // Insertion into QLDB
    txn.execute("INSERT INTO ExampleTable ?", ionStruct);

    // Fetching from QLDB
    Result result = txn.execute("SELECT * from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Map the fields of the Ion struct to Java values and construct a new POJO
        ionStruct = (IonStruct)ionValue;
        IonString exampleString = (IonString)ionStruct.get("ExampleString");
        IonInt exampleInt = (IonInt)ionStruct.get("ExampleInt");
        aPojo = new ExampleStruct(exampleString.stringValue(),
 exampleInt.intValue());
    }

    // examplePojo is now the document fetched from QLDB
    return aPojo;
});
```

Alternativ können Sie die Jackson-Bibliothek verwenden, um Datentypen zu und von Ion zuzuordnen. Die Bibliothek unterstützt die anderen Ion-Datentypen, aber dieses Beispiel konzentriert sich auf den`struct` Typ.

```
class ExampleStruct {
    public String exampleString;
```

```
    public int exampleInt;

    @JsonCreator
    public ExampleStruct(@JsonProperty("ExampleString") String exampleString,
                         @JsonProperty("ExampleInt") int exampleInt) {
        this.exampleString = exampleString;
        this.exampleInt = exampleInt;
    }

    @JsonProperty("ExampleString")
    public String getExampleString() {
        return this.exampleString;
    }

    @JsonProperty("ExampleInt")
    public int getExampleInt() {
        return this.exampleInt;
    }
}

// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();
// Instantiate an IonObjectMapper from the Jackson library
IonObjectMapper ionMapper = new IonValueMapper(ionSystem);

ExampleStruct examplePojo = driver.execute((txn) -> {
    // Transforming a POJO to Ion
    ExampleStruct aPojo = new ExampleStruct("Hello world!", 256);
    IonValue ionStruct;
    try  {
        // Use the mapper to convert Java objects into Ion
        ionStruct = ionMapper.writeValueAsIonValue(aPojo);
    } catch (IOException e) {
        // Wrap the exception and throw it for the sake of simplicity in this
 example
        throw new RuntimeException(e);
    }

    // Insertion into QLDB
    txn.execute("INSERT INTO ExampleTable ?", ionStruct);

    // Fetching from QLDB
    Result result = txn.execute("SELECT * from ExampleTable");
    // Assume there is only one document in ExampleTable
```

```
    for (IonValue ionValue : result)
    {
        // Use the mapper to convert Ion to Java objects
        try {
            aPojo = ionMapper.readValue(ionValue, ExampleStruct.class);
        } catch (IOException e) {
            // Wrap the exception and throw it for the sake of simplicity in this
 example
            throw new RuntimeException(e);
        }
    }

    // examplePojo is now the document fetched from QLDB
    return aPojo;
});
```

.NET

Verwenden Sie die Bibliothek Amazon.QLDB.Driver.Serialization, um native C#-Datentypen Ion zuzuordnen. Die Bibliothek unterstützt die anderen Ion-Datentypen, aber dieses Beispiel konzentriert sich auf den`struct` Typ.

```
using Amazon.QLDB.Driver.Generic;
using Amazon.QLDB.Driver.Serialization;
...

IAsyncQldbDriver driver = AsyncQldbDriver.Builder()
    .WithLedger("vehicle-registration")
    // Add Serialization library
    .WithSerializer(new ObjectSerializer())
    .Build();

// Creating a C# POCO.
ExampleStruct exampleStruct = new ExampleStruct
{
    ExampleString = "Hello world!",
    ExampleInt = 256
};

IAsyncResult<ExampleStruct> selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
```

```
    await txn.Execute(txn.Query<Document>("UPDATE ExampleTable SET ExampleStruct
 = ?", exampleStruct));

    // Fetching from QLDB.
    return await txn.Execute(txn.Query<ExampleStruct>("SELECT VALUE ExampleStruct
 from ExampleTable"));
});

await foreach (ExampleStruct row in selectResult)
{
    Console.WriteLine(row.ExampleString);
    Console.WriteLine(row.ExampleInt);
}
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

Alternativ hierzu können Sie auch Amazon verwenden. IonDotnet.Builders-Bibliothek zur Verarbeitung von Ion-Datentypen.

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Creating Ion struct.
IIonValue ionStruct = valueFactory.NewEmptyStruct();
ionStruct.SetField("ExampleString", valueFactory.NewString("Hello world!"));
ionStruct.SetField("ExampleInt", valueFactory.NewInt(256));

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleStruct = ?", ionStruct);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleStruct from ExampleTable");
});
```

```
string retrievedString = null;
int? retrievedInt = null;

await foreach (IIonValue ionValue in selectResult)
{
    retrievedString = ionValue.GetField("ExampleString").StringValue;
    retrievedInt = ionValue.GetField("ExampleInt").IntValue;
}
```

Go

```
exampleStruct, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aStruct := map[string]interface{} {
  "ExampleString": "Hello World!",
  "ExampleInt": 256,
 }

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleStruct = ?", aStruct)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleStruct FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult map[string]interface{}
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleStruct is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonStruct(driver: QldbDriver): Promise<any> {
    let exampleStruct: any = {stringValue: "Hello World!", intValue: 256};
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            // Inserting into QLDB
            await txn.execute("INSERT INTO ExampleTable ?", exampleStruct);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT * FROM
 ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // We can get all the keys of Ion struct and their associated values
            const ionFieldNames: string[] = ionValue.fieldNames();

            // Getting key and value of Ion struct to TypeScript String and Number
            const nativeStringVal: string =
 ionValue.get(ionFieldNames[0]).stringValue();
            const nativeIntVal: number =
 ionValue.get(ionFieldNames[1]).numberValue();
            // Alternatively, we can access to Ion struct fields, using their
 literal field names:
            //   const nativeStringVal = ionValue.get("stringValue").stringValue();
            //   const nativeIntVal = ionValue.get("intValue").numberValue();

            exampleStruct = {[ionFieldNames[0]]: nativeStringVal,
 [ionFieldNames[1]]: nativeIntVal};
            return exampleStruct;
        })
    );
}
```

Python

```
def update_and_query_ion_struct(txn):
    # QLDB can take in a Python struct
    a_struct = {"ExampleString": "Hello world!", "ExampleInt": 256}

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleStruct = ?", a_struct)
```

```
    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleStruct FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python struct is a child class of Python struct
        a_struct = ion_value

    # example_struct is now the value fetched from QLDB
    return a_struct


example_struct = driver.execute_lambda(lambda txn: update_and_query_ion_struct(txn))
```

Nullwerte und dynamische Typen

QLDB unterstützt offene Inhalte und erzwingt keine Schema- oder Datentypdefinitionen für Dokumentfelder. Sie können Ion-Nullwerte auch in einem QLDB-Dokument speichern. Alle vorherigen Beispiele gehen davon aus, dass jeder zurückgegebene Datentyp bekannt ist und nicht Null ist. Die folgenden Beispiele zeigen, wie Sie mit Ion arbeiten

Java

```
// Empty variables
String exampleString = null;
Integer exampleInt = null;

// Assume ionValue is some queried data from QLDB
IonValue ionValue = null;

// Check the value type and assign it to the variable if it is not null
if (ionValue.getType() == IonType.STRING) {
    if (ionValue.isNullValue()) {
        exampleString = null;
    } else {
        exampleString = ((IonString)ionValue).stringValue();
    }
} else if (ionValue.getType() == IonType.INT) {
    if (ionValue.isNullValue()) {
        exampleInt = null;
    } else {
        exampleInt = ((IonInt)ionValue).intValue();
```

```
    }
};

// Creating null values
IonSystem ionSystem = IonSystemBuilder.standard().build();

// A null value still has an Ion type
IonString ionString;
if (exampleString == null) {
    // Specifically a null string
    ionString = ionSystem.newNullString();
} else {
    ionString = ionSystem.newString(exampleString);
}

IonInt ionInt;
if (exampleInt == null) {
    // Specifically a null int
    ionInt = ionSystem.newNullInt();
} else {
    ionInt = ionSystem.newInt(exampleInt);
}

// Special case regarding null!
// There is a generic null type which has Ion type 'Null'.
// The above null values still have the types 'String' and 'Int'
IonValue specialNull = ionSystem.newNull();
if (specialNull.getType() == IonType.NULL) {
    // This is true!
}
if (specialNull.isNullValue()) {
    // This is also true!
}
if (specialNull.getType() == IonType.STRING || specialNull.getType() == IonType.INT)
 {
    // This is false!
}
```

.NET

```
// Empty variables.
string exampleString = null;
int? exampleInt = null;
```

```
// Assume ionValue is some queried data from QLDB.
IIonValue ionValue;

if (ionValue.Type() == IonType.String)
{
    exampleString = ionValue.StringValue;
}
else if (ionValue.Type() == IonType.Int)
{
    if (ionValue.IsNull)
    {
        exampleInt = null;
    }
    else
    {
        exampleInt = ionValue.IntValue;
    }
};

// Creating null values.
IValueFactory valueFactory = new ValueFactory();

// A null value still has an Ion type.
IIonValue ionString = valueFactory.NewString(exampleString);

IIonValue ionInt;
if (exampleInt == null)
{
    // Specifically a null int.
    ionInt = valueFactory.NewNullInt();
}
else
{
    ionInt = valueFactory.NewInt(exampleInt.Value);
}

// Special case regarding null!
// There is a generic null type which has Ion type 'Null'.
IIonValue specialNull = valueFactory.NewNull();
if (specialNull.Type() == IonType.Null) {
    // This is true!
}
if (specialNull.IsNull) {
```

```
    // This is also true!
}
```

Go

Einschränkungen beim Rangieren

In Go behaltennil Werte ihren Typ nicht bei, wenn Sie sie marshalisieren und dann wieder aufheben. Einnil Wert wechselt zu Ion Null, aber Ion Null demarshalliert eher zu einem Nullwert alsnil.

```
ionNull, err := ion.MarshalText(nil) // ionNull is set to ion null
if err != nil {
    return
}

var result int
err = ion.Unmarshal(ionNull, &result) // result unmarshals to 0
if err != nil {
    return
}
```

Node.js

```
// Empty variables
let exampleString: string;
let exampleInt: number;

// Assume ionValue is some queried data from QLDB
// Check the value type and assign it to the variable if it is not null
if (ionValue.getType() === IonTypes.STRING) {
    if (ionValue.isNull()) {
        exampleString = null;
    } else {
        exampleString = ionValue.stringValue();
    }
} else if (ionValue.getType() === IonTypes.INT) {
    if (ionValue.isNull()) {
        exampleInt = null;
    } else {
        exampleInt = ionValue.numberValue();
    }
}
```

```
// Creating null values
if (exampleString === null) {
    ionString = dom.load('null.string');
} else {
    ionString = dom.load.of(exampleString);
}

if (exampleInt === null) {
    ionInt = dom.load('null.int');
} else {
    ionInt = dom.load.of(exampleInt);
}

// Special case regarding null!
// There is a generic null type which has Ion type 'Null'.
// The above null values still have the types 'String' and 'Int'
specialNull: dom.Value = dom.load("null.null");
if (specialNull.getType() === IonType.NULL) {
    // This is true!
}
if (specialNull.getType() === IonType.STRING || specialNull.getType() ===
 IonType.INT) {
    // This is false!
}
```

Python

```
# Empty variables
example_string = None
example_int = None

# Assume ion_value is some queried data from QLDB
# Check the value type and assign it to the variable if it is not null
if ion_value.ion_type == IonType.STRING:
    if isinstance(ion_value, IonPyNull):
        example_string = None
    else:
        example_string = ion_value
elif ion_value.ion_type == IonType.INT:
    if isinstance(ion_value, IonPyNull):
        example_int = None
    else:
```

```
        example_int = ion_value

# Creating Ion null values
if example_string is None:
    # Specifically a null string
    ion_string = loads("null.string")
else:
    # QLDB can take in Python string
    ion_string = example_string
if example_int is None:
    # Specifically a null int
    ion_int = loads("null.int")
else:
    # QLDB can take in Python int
    ion_int = example_int


# Special case regarding null!
# There is a generic null type which has Ion type 'Null'.
# The above null values still have the types 'String' and 'Int'
special_null = loads("null.null")
if special_null.ion_type == IonType.NULL:
    # This is true!
if special_null.ion_type == IonType.STRING or special_null.ion_type == IonType.INT:
    # This is false!
```

Downkonvertierung zu JSON

Wenn Ihre Anwendung JSON-Kompatibilität erfordert, können Sie Amazon Ion-Daten in JSON herunterkonvertieren. In bestimmten Fällen, in denen Ihre Daten die reichhaltigen Ionentypen verwenden, die in JSON nicht existieren, ist die Konvertierung von Ion in JSON jedoch mit Verlusten verbunden.

Einzelheiten zu den Konvertierungsregeln von Ion in JSON finden Sie im Amazon Ion-Cookbook unter Down-Converting to JSON.

Arbeiten mit Daten und Historie in Amazon QLDB

Die folgenden Themen enthalten grundlegende Beispiele für CRUD-Befehle (erstellen/aktualisieren/aktualisieren/aktualisieren/löschen). Sie können diese Anweisungen manuell ausführen, indem Sie den PartiQL-Editor auf der QLDB-Konsole oder die QLDB-Shell verwenden. Dieser Leitfaden führt Sie auch durch den Prozess, wie QLDB mit Ihren Daten umgeht, wenn Sie Änderungen an Ihrem Ledger vornehmen.

QLDB unterstützt die PartiQL-Abfragesprache.

Codebeispiele, die zeigen, wie ähnliche Anweisungen mithilfe des QLDB-Treibers programmgesteuert ausgeführt werden, finden Sie in den Tutorials unterErste Schritte mit dem Treiber.

> **Tip**
>
> Im Folgenden finden Sie eine kurze Zusammenfassung der Tipps und bewährten Methoden für die Arbeit mit PartiQL in QLDB:
>
> - Machen Sie sich mit Parallelität und Transaktionslimits vertraut — Alle Anweisungen, einschließlich`SELECT` Abfragen, unterliegen Optimist Concurrency Control (OCC) - Konflikten und Transaktionslimits, einschließlich eines Transaktions-Timeouts von 30 Sekunden.
> - Indizes verwenden — Verwenden Sie Indizes mit hoher Kardinalität und führen Sie gezielte Abfragen aus, um Ihre Aussagen zu optimieren und vollständige Tabellenscans zu vermeiden. Weitere Informationen hierzu finden Sie unter Optimieren der Abfrageleistung.
> - Verwenden Sie Gleichheitsprädikate — Indexierte Suchvorgänge erfordern einen Gleichheitsoperator (`=`oder`IN`). Ungleichheitsoperatoren (`<>`,`LIKE`,`BETWEEN`) eignen sich nicht für indexierte Suchvorgänge und führen zu vollständigen Tabellenscans.
> - Nur innere Joins verwenden — QLDB unterstützt nur innere Joins. Es hat sich bewährt, Felder miteinander zu verknüpfen, die für jede Tabelle, die Sie verknüpfen, indexiert sind. Wählen Sie Indizes mit hoher Kardinalität sowohl für die Verbindungskriterien als auch für die Gleichheitsprädikate.

Themen

- Abfragen Ihrer Daten
- Metadaten von Dokumenten abfragen
- Verwenden der BY-Klausel zur Abfrage der Dokument-ID
- Dokumente aktualisieren und löschen
- Abfragen des Revisionsverlaufs
- Redigieren von Dokumentrevisionen
- Optimieren der Abfrageleistung
- Abrufen von PartiQL-Anweisungsstatistiken
- Abfragen des Systemkatalogs
- Verwalten von Tabellen
- Verwalten von Indexen
- Eindeutige IDs in Amazon QLDB

Tabellen mit Indizes erstellen und Dokumente einfügen

Nachdem Sie ein Amazon QLDB-Ledger erstellt haben, besteht Ihr erster Schritt darin, eine Tabelle mit einer grundlegendenCREATE TABLE Aussage zu erstellen. Tabellen bestehen ausQLDB-Dokumente, wobei es sich um Datensätze im Amazon`struct` Ion-Format handelt.

Themen

- Erstellen von Tabellen und Indizes
- Einfügen von Dokumenten

Erstellen von Tabellen und Indizes

Tabellen haben einfache Namen ohne Namespaces, bei denen zwischen Groß- und Kleinschreibung unterschieden wird. QLDB unterstützt offene Inhalte und erzwingt kein Schema, sodass Sie beim Erstellen von Tabellen keine Attribute oder Datentypen definieren.

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle
```

Eine`CREATE TABLE` Anweisung gibt die vom System zugewiesene ID der neuen Tabelle zurück. Alle vom System zugewiesenen IDs in QLDB sind universell eindeutige Identifikatoren (UUID), die jeweils in einer Base62-kodierten Zeichenfolge dargestellt werden.

> **Note**
>
> Optional können Sie Tags für eine Tabellenressource definieren, während Sie die Tabelle erstellen. Um zu erfahren wie dies geht, vgl. Markierung von Tabellen bei der Erstellung.

Sie können auch Indizes für Tabellen erstellen, um die Abfrageleistung zu optimieren.

```
CREATE INDEX ON VehicleRegistration (VIN)
```

```
CREATE INDEX ON VehicleRegistration (LicensePlateNumber)
```

```
CREATE INDEX ON Vehicle (VIN)
```

> **Important**
>
> QLDB benötigt einen Index, um ein Dokument effizient nachschlagen zu können. Ohne Index muss QLDB beim Lesen von Dokumenten einen vollständigen Tabellenscan durchführen. Dies kann bei großen Tabellen zu Leistungsproblemen führen, einschließlich Parallelitätskonflikten und Transaktions-Timeouts.
>
> Um Tabellenscans zu vermeiden, müssen Sie Anweisungen mit einer`WHERE` Prädikatklausel unter Verwendung eines Gleichheitsoperators (=oder`IN`) für ein indiziertes Feld oder eine Dokument-ID ausführen. Weitere Informationen finden Sie unter Optimieren der Abfrageleistung.

Beachten Sie beim Erstellen von Indizes die folgenden Einschränkungen:

- Ein Index kann nur für ein einzelnes Feld der obersten Ebene erstellt werden. Zusammengesetzte, verschachtelte, eindeutige und funktionsbasierte Indizes werden nicht unterstützt.
- Sie können einen Index für alle Ion-Datentypen erstellen, einschließlich`list` und`struct`. Sie können die indizierte Suche jedoch nur nach Gleichheit des gesamten Ionenwerts durchführen,

unabhängig vom Ionentyp. Wenn Sie beispielsweise einen`list` Typ als Index verwenden, können Sie keine indizierte Suche nach einem Element in der Liste durchführen.

- Die Abfrageleistung wird nur verbessert, wenn Sie ein Gleichheitsprädikat verwenden, z. B.`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`.

  QLDB berücksichtigt keine Ungleichungen in Abfrageprädikaten. Daher werden bereichsgefilterte Scans nicht implementiert.
- Bei Namen von indizierten Feldern muss zwischen Groß- und Kleinschreibung unterschieden werden. Sie dürfen höchstens 128 Zeichen lang sein.
- Die Indexerstellung in QLDB erfolgt asynchron. Die Zeit, die zum Erstellen eines Index für eine nicht leere Tabelle benötigt wird, hängt von der Tabellengröße ab. Weitere Informationen finden Sie unter Verwalten von Indexen.

Einfügen von Dokumenten

Anschließend können Sie Dokumente in Ihre Tabellen einfügen. QLDB-Dokumente werden im Amazon Ion-Format gespeichert. Die folgenden PartiQL-INSERT-Anweisungen enthalten eine Teilmenge der Beispieldaten für die Fahrzeugzulassung, die in Erste Schritte mit der Amazon QLDB-Konsole verwendet werden.

```
INSERT INTO VehicleRegistration
<< {
    'VIN' : '1N4AL11D75C109151',
    'LicensePlateNumber' : 'LEWISR261LL',
    'State' : 'WA',
    'City' : 'Seattle',
    'PendingPenaltyTicketAmount' : 90.25,
    'ValidFromDate' : `2017-08-21T`,
    'ValidToDate' : `2020-05-11T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId' : '294jJ3YUoH1IEEm8GSabOs' },
        'SecondaryOwners' : [ { 'PersonId' : '5Ufgdlnj06gF5CWcOIu64s' } ]
    }
},
{
    'VIN' : 'KM8SRDHF6EU074761',
    'LicensePlateNumber' : 'CA762X',
    'State' : 'WA',
    'City' : 'Kent',
```

```
    'PendingPenaltyTicketAmount' : 130.75,
    'ValidFromDate' : `2017-09-14T`,
    'ValidToDate' : `2020-06-25T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': 'IN7MvYtUjkp1GMZu0F6CG9' },
        'SecondaryOwners' : []
    }
} >>
```

```
INSERT INTO Vehicle
<< {
    'VIN' : '1N4AL11D75C109151',
    'Type' : 'Sedan',
    'Year' : 2011,
    'Make' : 'Audi',
    'Model' : 'A5',
    'Color' : 'Silver'
} ,
{
    'VIN' : 'KM8SRDHF6EU074761',
    'Type' : 'Sedan',
    'Year' : 2015,
    'Make' : 'Tesla',
    'Model' : 'Model S',
    'Color' : 'Blue'
} >>
```

PartiQL-Syntax und -Semantik

- Feldnamen werden in einfache Anführungszeichen eingeschlossen (`'...'`).
- Zeichenfolgenwerte werden ebenfalls in einfache Anführungszeichen eingeschlossen (`'...'`).
- Zeitstempel werden in umgekehrten Anzeichen (`` `...` ``) eingeschlossen. Mit umgekehrten Anführungszeichen können Sie beliebige Ion-Literalwerte kennzeichnen.
- Ganzzahlen und Dezimalstellen sind Literalwerte, die nicht bezeichnet werden müssen.

Weitere Informationen zur Syntax und Semantik von PartiQL finden Sie unterIon mit PartiQL in Amazon QLDB abfragen.

Eine `INSERT`-Anweisung erstellt die anfängliche Revision eines Dokuments mit einer Versionsnummer von Null. Um jedes Dokument eindeutig zu identifizieren, weist QLDB als Teil der

Metadaten eine Dokument-ID zu. Insert-Anweisungen geben die ID jedes eingefügten Dokuments zurück.

> **Important**
>
> Da QLDB kein Schema erzwingt, können Sie dasselbe Dokument mehrmals in eine Tabelle einfügen. Jede Einfügeanweisung überträgt einen separaten Dokumenteintrag in das Journal, und QLDB weist jedem Dokument eine eindeutige ID zu.

Informationen zum Abfragen der Dokumente, die Sie in Ihre Tabelle eingefügt haben, finden Sie unter Abfragen Ihrer Daten.

Abfragen Ihrer Daten

In der Benutzeransicht wird nur die letzte, nicht gelöschte Version Ihrer Benutzerdaten angezeigt. Dies ist die Standardansicht in Amazon QLDB. Dies bedeutet, dass keine speziellen Kriterien erforderlich sind, wenn Sie nur Ihre Daten abfragen möchten.

Einzelheiten zur Syntax und zu den Parametern der folgenden Abfragebeispiele finden SieSELECT in der Amazon QLDB PartiQL-Referenz.

Themen

- Grundlegende Abfragen
- Projektionen und Filter
- Joins
- Nested data

Grundlegende Abfragen

Grundlegende `SELECT`-Abfragen geben die Dokumente zurück, die Sie in die Tabelle eingefügt haben.

> **Warning**
>
> Wenn Sie eine Abfrage in QLDB ohne indizierte Suche ausführen, wird ein vollständiger Tabellenscan aufgerufen. PartiQL unterstützt solche Abfragen, da es SQL-kompatibel ist. Führen Sie jedoch keine Tabellenscans für produktive Anwendungsfälle in QLDB aus.

Tabellenscans können bei großen Tabellen zu Leistungsproblemen führen, einschließlich Parallelitätskonflikten und Transaktions-Timeouts.
Um Tabellenscans zu vermeiden, müssen Sie Anweisungen mit einer`WHERE` Prädikatklausel ausführen, die einen Gleichheitsoperator für ein indiziertes Feld oder eine Dokument-ID verwenden, z. B.`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`. Weitere Informationen finden Sie unter Optimieren der Abfrageleistung.

Die folgenden Abfragen zeigen Ergebnisse für die Fahrzeugscheine, die Sie zuvor eingegeben habenTabellen mit Indizes erstellen und Dokumente einfügen. Die Reihenfolge der Ergebnisse ist nicht spezifisch und kann für jede`SELECT` Abfrage variieren. Sie sollten sich bei keiner Abfrage in QLDB auf die Reihenfolge der Ergebnisse verlassen.

```
SELECT * FROM VehicleRegistration
WHERE LicensePlateNumber IN ('LEWISR261LL', 'CA762X')
```

```
{
    VIN: "1N4AL11D75C109151",
    LicensePlateNumber: "LEWISR261LL",
    State: "WA",
    City: "Seattle",
    PendingPenaltyTicketAmount: 90.25,
    ValidFromDate: 2017-08-21T,
    ValidToDate: 2020-05-11T,
    Owners: {
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    }
},
{
    VIN: "KM8SRDHF6EU074761",
    LicensePlateNumber: "CA762X",
    State: "WA",
    City: "Kent",
    PendingPenaltyTicketAmount: 130.75,
    ValidFromDate: 2017-09-14T,
    ValidToDate: 2020-06-25T,
    Owners: {
        PrimaryOwner: { PersonId: "IN7MvYtUjkp1GMZu0F6CG9" },
        SecondaryOwners: []
    }
```

```
}
```

```
SELECT * FROM Vehicle
WHERE VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

```
{
    VIN: "1N4AL11D75C109151",
    Type: "Sedan",
    Year: 2011,
    Make: "Audi",
    Model: "A5",
    Color: "Silver"
},
{
    VIN: "KM8SRDHF6EU074761",
    Type: "Sedan",
    Year: 2015,
    Make: "Tesla",
    Model: "Model S",
    Color: "Blue"
}
```

Important

In PartiQL verwenden Sie einfache Anführungszeichen, um Zeichenfolgen in der Data Manipulation Language (DML) oder in Abfrageanweisungen zu bezeichnen. Die QLDB-Konsole und die QLDB-Shell geben Abfrageergebnisse jedoch im Amazon Ion-Textformat zurück, sodass Sie Zeichenfolgen in doppelten Anführungszeichen sehen.
Diese Syntax ermöglicht es der PartiQL-Abfragesprache, die SQL-Kompatibilität aufrechtzuerhalten, und dem Amazon Ion-Textformat, um die JSON-Kompatibilität aufrechtzuerhalten.

Projektionen und Filter

Sie können Projektionen (gezielt`SELECT`) und andere Standardfilter (`WHERE`Klauseln) durchführen. Die folgende Abfrage gibt eine Teilmenge der Dokumentfelder aus der `VehicleRegistration`-Tabelle zurück. Sie filtert nach Fahrzeugen mit den folgenden Kriterien:

- Zeichenkettenfilter — Er ist in Seattle registriert.

- Dezimalfilter — Der Betrag eines ausstehenden Straftickets ist geringer als`100.0`.
- Datumsfilter — Es hat ein Registrierungsdatum, das am oder nach dem 4. September 2019 gültig ist.

```
SELECT r.VIN, r.PendingPenaltyTicketAmount, r.Owners
FROM VehicleRegistration AS r
WHERE r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
AND r.City = 'Seattle' --string
AND r.PendingPenaltyTicketAmount < 100.0 --decimal
AND r.ValidToDate >= `2019-09-04T` --timestamp with day precision
```

```
{
    VIN: "1N4AL11D75C109151",
    PendingPenaltyTicketAmount: 90.25,
    Owners: {
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    }
}
```

Joins

Sie können auch innere Join-Abfragen schreiben. Das folgende Beispiel zeigt eine implizite innere Join-Abfrage, die alle Zulassungsdokumente zusammen mit den Attributen der registrierten Fahrzeuge zurückgibt.

```
SELECT * FROM VehicleRegistration AS r, Vehicle AS v
WHERE r.VIN = v.VIN
AND r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

```
{
    VIN: "1N4AL11D75C109151",
    LicensePlateNumber: "LEWISR261LL",
    State: "WA",
    City: "Seattle",
    PendingPenaltyTicketAmount: 90.25,
    ValidFromDate: 2017-08-21T,
    ValidToDate: 2020-05-11T,
    Owners: {
```

```
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    },
    Type: "Sedan",
    Year: 2011,
    Make: "Audi",
    Model: "A5",
    Color: "Silver"
},
{
    VIN: "KM8SRDHF6EU074761",
    LicensePlateNumber: "CA762X",
    State: "WA",
    City: "Kent",
    PendingPenaltyTicketAmount: 130.75,
    ValidFromDate: 2017-09-14T,
    ValidToDate: 2020-06-25T,
    Owners: {
        PrimaryOwner: { PersonId: "IN7MvYtUjkp1GMZu0F6CG9" },
        SecondaryOwners: []
    },
    Type: "Sedan",
    Year: 2015,
    Make: "Tesla",
    Model: "Model S",
    Color: "Blue"
}
```

Oder Sie können dieselbe innere Join-Abfrage in der folgenden expliziten Syntax schreiben.

```
SELECT * FROM VehicleRegistration AS r INNER JOIN Vehicle AS v
ON r.VIN = v.VIN
WHERE r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Nested data

Sie können PartiQL in QLDB verwenden, um verschachtelte Daten in Dokumenten abzufragen. Das folgende Beispiel zeigt eine korrelierte Unterabfrage, die verschachtelte Daten vereinfacht. Das Zeichen `@` ist hier technisch optional. Es gibt jedoch explizit an, dass Sie die `Owners`-Struktur innerhalb von `VehicleRegistration` haben möchten, und nicht eine andere Sammlung mit dem Namen `Owners` (sofern vorhanden).

```
SELECT
    r.VIN,
    o.SecondaryOwners
FROM
    VehicleRegistration AS r, @r.Owners AS o
WHERE
    r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

```
{
    VIN: "1N4AL11D75C109151",
    SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
},
{
    VIN: "KM8SRDHF6EU074761",
    SecondaryOwners: []
}
```

Das folgende Beispiel zeigt eine Unterabfrage in der SELECT-Liste, die verschachtelte Daten projiziert, zusätzlich zu einem inneren Join.

```
SELECT
    v.Make,
    v.Model,
    (SELECT VALUE o.PrimaryOwner.PersonId FROM @r.Owners AS o) AS PrimaryOwner
FROM
    VehicleRegistration AS r, Vehicle AS v
WHERE
    r.VIN = v.VIN AND r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

```
{
    Make: "Audi",
    Model: "A5",
    PrimaryOwner: ["294jJ3YUoH1IEEm8GSabOs"]
},
{
    Make: "Tesla",
    Model: "Model S",
    PrimaryOwner: ["IN7MvYtUjkp1GMZu0F6CG9"]
}
```

Die folgende Abfrage gibt die`PersonId` und Indexnummer (Ordinalzahl) jeder Person in der`Owners.SecondaryOwners` Liste für ein`VehicleRegistration` Dokument zurück.

```
SELECT s.PersonId, owner_idx
FROM VehicleRegistration AS r, @r.Owners.SecondaryOwners AS s AT owner_idx
WHERE r.VIN = '1N4AL11D75C109151'
```

```
{
    PersonId: "5Ufgdlnj06gF5CWcOIu64s",
    owner_idx: 0
}
```

Weitere Informationen zur Abfrage Ihrer Dokumentmetadaten finden Sie unter Metadaten von Dokumenten abfragen.

Metadaten von Dokumenten abfragen

Eine `INSERT`-Anweisung erstellt die anfängliche Revision eines Dokuments mit einer Versionsnummer von Null. Um jedes Dokument eindeutig zu identifizieren, weist Amazon QLDB als Teil der Metadaten eine Dokument-ID zu.

Neben der Dokument-ID und der Versionsnummer speichert QLDB weitere systemgenerierte Metadaten für jedes Dokument in einer Tabelle. Diese Metadaten umfassen Transaktionsinformationen, Journalattribute und den Hashwert des Dokuments.

Alle vom System zugewiesenen IDs sind universell eindeutige Identifikatoren (UUID), die jeweils in einer Base62-kodierten Zeichenfolge dargestellt werden. Weitere Informationen finden Sie unter Eindeutige IDs in Amazon QLDB.

Themen

- Engagierter Standpunkt
- Vereinheitlichung der Ansichten von Engagierten und Benutzern

Engagierter Standpunkt

Sie können auf Dokumentmetadaten zugreifen, indem Sie die übergebene Ansicht abfragen. Diese Ansicht gibt Dokumente aus der vom System definierten Tabelle zurück, die direkt Ihrer Benutzertabelle entspricht. Es enthält die letzte übernommene, nicht gelöschte Version Ihrer Daten

und der vom System generierten Metadaten. Zum Abfragen dieser Ansicht fügen Sie das Präfix `_ql_committed_` zum Tabellennamen in Ihrer Abfrage hinzu. (Das Präfix`_ql_` ist in QLDB für Systemobjekte reserviert.)

```
SELECT * FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Unter Verwendung der zuvorTabellen mit Indizes erstellen und Dokumente einfügen eingegebenen Daten zeigt die Ausgabe dieser Abfrage den Systeminhalt der letzten Version jedes nicht gelöschten Dokuments. Das Systemdokument enthält Metadaten, die im`metadata` Feld verschachtelt sind, und Ihre Benutzerdaten sind in dem`data` Feld verschachtelt.

```
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:14
    },
    hash:{{wCsmM6qD4STxz0WYmE+47nZvWtcCz9D6zNtCiM5GoWg=}},
    data:{
        VIN: "1N4AL11D75C109151",
        LicensePlateNumber: "LEWISR261LL",
        State: "WA",
        City: "Seattle",
        PendingPenaltyTicketAmount: 90.25,
        ValidFromDate: 2017-08-21T,
        ValidToDate: 2020-05-11T,
        Owners: {
            PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
            SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
        }
    },
    metadata:{
        id:"3Qv67yjXEwB9SjmvkuG6Cp",
        version:0,
        txTime:2019-06-05T20:53:321d-3Z,
        txId:"HgXAkLjAtV0HQ4lNYdzX60"
    }
},
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:14
```

```
    },
    hash:{{wPuwH60TtcCvg/23BFp+redRXuCALkbDihkEvCX22Jk=}},
    data:{
        VIN: "KM8SRDHF6EU074761",
        LicensePlateNumber: "CA762X",
        State: "WA",
        City: "Kent",
        PendingPenaltyTicketAmount: 130.75,
        ValidFromDate: 2017-09-14T,
        ValidToDate: 2020-06-25T,
        Owners: {
            PrimaryOwner: { PersonId: "IN7MvYtUjkp1GMZu0F6CG9" },
            SecondaryOwners: []
        }
    },
    metadata:{
        id:"JOzfB3lWqGU727mpPeWyxg",
        version:0,
        txTime:2019-06-05T20:53:321d-3Z,
        txId:"HgXAkLjAtV0HQ4lNYdzX60"
    }
}
```

Festgelegte Sichtfelder

- `blockAddress`— Die Position des Blocks im Journal Ihres Hauptbuchs, in dem die Überarbeitung des Dokuments bestätigt wurde. Eine Adresse, die für die kryptografische Verifizierung verwendet werden kann, hat die folgenden zwei Felder.
  - `strandId`— Die eindeutige ID des Journalstrangs, der den Block enthält.
  - `sequenceNo`— Eine Indexnummer, die die Position des Blocks innerhalb des Strangs angibt.

> **Note**
>
> Beide Dokumente in diesem Beispiel haben eine identische `blockAddress` mit der gleichen `sequenceNo`. Da diese Dokumente in einer einzelnen Transaktion eingefügt wurden (und in diesem Fall in einer einzelnen Anweisung), werden sie im gleichen Block festgeschrieben.

- `hash`— Der SHA-256-Ionen-Hashwert, der die Dokumentenversion eindeutig darstellt. Der Hash deckt die Revisionen`data` und`metadata` Felder ab und kann für die kryptografische Überprüfung verwendet werden.
- `data`— Die Benutzerdatenattribute des Dokuments.

  Wenn Sie eine Revision redigieren, wird diese`data` Struktur durch ein`dataHash` Feld ersetzt, dessen Wert der Ionen-Hash der entfernten`data` Struktur ist.
- `metadata`— Die Metadatenattribute des Dokuments.
  - `id`— Die vom System zugewiesene eindeutige ID des Dokuments.
  - `version`— Die Versionsnummer des Dokuments. Dies ist eine auf Null basierende Ganzzahl, die mit jeder Dokumentversion inkrementiert wird.
  - `txTime`— Der Zeitstempel, zu dem die Dokumentenrevision an das Journal übermittelt wurde.
  - `txId`— Die eindeutige ID der Transaktion, die die Dokumentaktualisierung durchgeführt hat.

Vereinheitlichung der Ansichten von Engagierten und Benutzern

Sie können Abfragen schreiben, die eine Tabelle in der Committe-Ansicht mit einer Tabelle in der Benutzeransicht verknüpfen. Beispielsweise möchten Sie vielleicht das Dokument`id` einer Tabelle mit einem benutzerdefinierten Feld einer anderen Tabelle verknüpfen.

Die folgende Abfrage verknüpft zwei Tabellen, die benannt sind`DriversLicense` und`Person` in ihren`PersonId` jeweiligen`id` Dokumentfeldern benannt sind, wobei für letztere die Committ-Ansicht verwendet wird.

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS p
ON d.PersonId = p.metadata.id
WHERE p.metadata.id = '1CWScY2qHYI9G88C2SjvtH'
```

Informationen zur Abfrage des Dokument-ID-Felds in der Standard-Benutzeransicht finden Sie unter Verwenden der BY-Klausel zur Abfrage der Dokument-ID.

Verwenden der BY-Klausel zur Abfrage der Dokument-ID

Sie können zwar Felder definieren, die als eindeutige Identifikatoren gedacht sind (z. B. die Fahrgestellnummer eines Fahrzeugs), die wahre eindeutige Kennung eines Dokuments ist jedoch das`id` Metadatenfeld, wie unter beschriebenEinfügen von Dokumenten. Aus diesem Grund können Sie das `id`-Feld zum Erstellen von Beziehungen zwischen Tabellen verwenden.

Auf das Dokument-`id`-Feld kann nur in der Committed-Ansicht direkt zugegriffen werden. Sie können es jedoch auch in der Standard-Benutzeransicht projizieren, indem Sie die `BY`-Klausel verwenden. Ein Beispiel finden Sie in der folgenden Abfrage und ihren Ergebnissen.

```
SELECT r_id, r.VIN, r.LicensePlateNumber, r.State, r.City, r.Owners
FROM VehicleRegistration AS r BY r_id
WHERE r_id = '3Qv67yjXEwB9SjmvkuG6Cp'
```

```
{
    r_id: "3Qv67yjXEwB9SjmvkuG6Cp",
    VIN: "1N4AL11D75C109151",
    LicensePlateNumber: "LEWISR261LL",
    State: "WA",
    City: "Seattle",
    Owners: {
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    }
}
```

In dieser Abfrage ist `r_id` ein vom Benutzer definierter Alias, der in der `FROM`-Klausel deklariert wird, mit dem `BY`-Schlüsselwort. Dieser `r_id`-Alias ist mit dem `id`-Metadatenattribut für jedes Dokument in der Ergebnisgruppe der Abfrage verbunden. Sie können diesen Alias in der `SELECT`-Klausel und der `WHERE`-Klausel einer Abfrage in der Benutzeransicht verwenden.

Für den Zugriff auf andere Metadatenattribute müssen Sie jedoch die bestätigte Ansicht abfragen.

Beitritt mit Dokument-ID

Angenommen, Sie verwenden das Dokument`id` einer Tabelle als Fremdschlüssel in einem benutzerdefinierten Feld einer anderen Tabelle. Sie können die`BY` Klausel verwenden, um eine innere Join-Abfrage für die beiden Tabellen in diesen Feldern zu schreiben (ähnlich wieVereinheitlichung der Ansichten von Engagierten und Benutzern im vorherigen Thema).

Im folgenden Beispiel werden zwei Tabellen, die benannt sind`DriversLicense` und`Person` in ihren`id` Feldern`PersonId` und Dokumenten benannt sind, verknüpft, wobei die`BY` Klausel für letzteres verwendet wird.

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
```

```
ON d.PersonId = pid
WHERE pid = '1CWScY2qHYI9G88C2SjvtH'
```

Wie Sie Änderungen an einem Dokument in Ihrer Tabelle vornehmen können, erfahren Sie unter Dokumente aktualisieren und löschen.

Dokumente aktualisieren und löschen

In Amazon QLDB ist eine Dokumentrevision eine Amazon Ion-Struktur, die eine einzelne Version einer Sequenz von Dokumenten darstellt, die durch eine eindeutige Dokument-ID identifiziert werden. Jede Revision enthält den vollständigen Datensatz des Dokuments, einschließlich Ihrer Benutzerdaten und systemgenerierter Metadaten. Jede Revision wird eindeutig durch eine Kombination aus Dokument-ID und nullbasierter Versionsnummer identifiziert.

Wenn Sie ein Dokument aktualisieren, erstellt QLDB eine neue Revision mit derselben Dokument-ID und einer inkrementierten Versionsnummer. Der Lebenszyklus eines Dokuments endet, wenn Sie es aus einer Tabelle löschen. Das bedeutet, dass keine Dokumentrevision mit derselben Dokument-ID erneut erstellt werden kann.

Änderungen an Dokumenten vornehmen

Beispiel: Die folgenden Anweisungen fügen eine neue Fahrzeugzulassung ein, aktualisieren die Stadt der Zulassung und löschen die Zulassung. Dies führt zu drei Revisionen eines Dokuments.

```
INSERT INTO VehicleRegistration
{
    'VIN' : '1HVBBAANXWH544237',
    'LicensePlateNumber' : 'LS477D',
    'State' : 'WA',
    'City' : 'Tacoma',
    'PendingPenaltyTicketAmount' : 42.20,
    'ValidFromDate' : `2011-10-26T`,
    'ValidToDate' : `2023-09-25T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': 'KmA3XPKKFqYCP2zhR3d0Ho' },
        'SecondaryOwners' : []
    }
}
```

> **Note**
>
> Insert-Anweisungen und andere DML-Anweisungen geben die ID jedes betroffenen Dokuments zurück. Bevor Sie fortfahren, speichern Sie diese ID, da Sie sie für die Verlaufsfunktion im nächsten Thema benötigen. Sie können die Dokument-ID auch mit der folgenden Abfrage finden.
>
> ```
> SELECT r_id FROM VehicleRegistration AS r BY r_id
> WHERE r.VIN = '1HVBBAANXWH544237'
> ```

```
UPDATE VehicleRegistration AS r
SET r.City = 'Bellevue'
WHERE r.VIN = '1HVBBAANXWH544237'
```

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

Weitere Beispiele und Informationen zur Syntax dieser DML-Anweisungen finden Sie unterUPDATE undDELETE in der Amazon QLDB PartiQL-Referenz.

Um bestimmte Elemente in ein Dokument einzufügen und zu entfernen, können SieUPDATE Anweisungen oder andere DML-Anweisungen verwenden, die mit demFROM Schlüsselwort beginnen. Weitere Informationen und Beispiele finden Sie in derVON (INSERT, REMOVE oder SET) Referenz.

Nachdem Sie ein Dokument gelöscht haben, können Sie es nicht mehr in der bestätigten oder Benutzeransicht abfragen. Weitere Informationen zur Abfrage des Versionsverlaufs dieses Dokuments mithilfe der integrierten Verlaufsfunktion finden Sie unter Abfragen des Revisionsverlaufs.

Abfragen des Revisionsverlaufs

Amazon QLDB speichert den vollständigen Verlauf jedes Dokuments in einer Tabelle. Sie können mithilfe einer Abfrage der integrierten Verlaufsfunktion alle drei Revisionen des Kfz-Zulassungsdokuments anzeigen, die Sie zuvor in Dokumente aktualisieren und löschen eingefügt, aktualisiert oder gelöscht haben.

Themen

Verlaufsfunktion

Die History-Funktion in QLDB ist eine PartiQL-Erweiterung, die Revisionen aus der systemdefinierten Ansicht Ihrer Tabelle zurückgibt. Sie enthält Ihre Daten und die zugehörigen Metadaten in demselben Schema wie die bestätigte Ansicht.

Syntax

```
SELECT * FROM history( table_name | 'table_id' [, `start-time` [, `end-time` ] ] ) AS h
[ WHERE h.metadata.id = 'id' ]
```

Argumente

Tabellenname | '***Tabellen-ID***'

Entweder der Tabellenname oder die Tabellen-ID. Ein Tabellenname ist eine PartiQL-Kennung, die Sie mit doppelten Anführungszeichen oder keinen Anführungszeichen bezeichnen können. Eine Tabellen-ID ist ein Zeichenfolgenliteralwert, der in einfache Anführungszeichen eingeschlossen werden muss. Weitere Informationen zur Verwendung von Tabellen-IDs finden Sie unterDen Verlauf inaktiver Tabellen abfragen.

`` `***Startzeit***` ``, `` `***Endzeit***` ``

(Optional) Gibt den Zeitraum an, in dem alle Revisionen aktiv waren. Diese Parameter geben nicht den Zeitraum an, in dem Revisionen im Rahmen einer Transaktion in das Journal übernommen wurden.

Die Start- und Endzeiten sind Ion-Timestamp-Literale, die mit backticks (`` `...` ``) gekennzeichnet werden können. Weitere Informationen hierzu finden Sie unter Ion mit PartiQL in Amazon QLDB abfragen.

Diese Zeitparameter haben das folgende Verhalten:

- Die Start - und Endzeit sind beide inklusive. Sie müssen im ISO 8601-Datums- und Zeitformat und in koordinierter Weltzeit (Coordinated Universal Time, UTC) angegeben werden.
- Die start-time (Anfangszeit) muss früher oder gleich der end-time (Endzeit) sein und kann jede beliebige Datum in der Vergangenheit sein.

- Die end-time (Endzeit) muss früher oder gleich dem aktuellen UTC-Datum und der Uhrzeit sein.
- Wenn Sie eine Startzeit, aber nicht eine Endzeit angeben, verwendet Ihre Abfrage das aktuelle Datum und die aktuelle Uhrzeit als Endzeit. Wenn Sie keine dieser Zeiten angeben, gibt die Abfrage den gesamten Verlauf zurück.

'*ID*'

(Optional) Die Dokument-ID, für die Sie den Revisionsverlauf abfragen möchten, gekennzeichnet durch einfache Anführungszeichen.

> **Tip**
>
> Es hat sich bewährt, eine Verlaufsabfrage sowohl mit einem Datumsbereich (Start- und Endzeit) als auch mit einer Dokument-ID (`metadata.id`) zu qualifizieren. In QLDB wird jede`SELECT` Anfrage in einer Transaktion verarbeitet und unterliegt einem Transaktions-Timeout-Limit.
> Verlaufsabfragen verwenden nicht die Indizes, die Sie für eine Tabelle erstellen. Der QLDB-Verlauf wird nur nach der Dokument-ID indexiert, und Sie können derzeit keine zusätzlichen Verlaufsindizes erstellen. Verlaufsabfragen, die eine Start- und Endzeit enthalten, profitieren von der Qualifizierung des Datumsbereichs.

Beispiel für die Verlaufsabfrage

Um die Historie des Fahrzeugscheins abzufragen, verwenden Sie den`id`, den Sie zuvor gespeichert habenDokumente aktualisieren und löschen. Die folgende Verlaufsabfrage gibt beispielsweise alle Revisionen für die Dokument-ID zurück`ADR2Ll1fGsU4Jr4EqTdnQF`, die jemals zwischen`2019-06-05T00:00:00Z` und aktiv waren`2019-06-05T23:59:59Z`.

> **Note**
>
> Denken Sie daran, dass die Start- und Endzeitparameter nicht den Zeitraum angeben, in dem Revisionen in einer Transaktion in das Journal übernommen wurden. Wenn beispielsweise eine Revision zuvor bestätigt wurde`2019-06-05T00:00:00Z` und nach dieser Startzeit aktiv war, gibt diese Beispielabfrage diese Revision in den Ergebnissen zurück.

Stellen Sie sicher, die `id`, die Start- und Endzeit durch Ihre eigenen Werte zu ersetzen.

```
SELECT * FROM history(VehicleRegistration, `2019-06-05T00:00:00Z`,
 `2019-06-05T23:59:59Z`) AS h
WHERE h.metadata.id = 'ADR2Ll1fGsU4Jr4EqTdnQF' --replace with your id
```

Ihre Abfrageergebnisse sollten wie folgt aussehen.

```
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:14
    },
    hash:{{B2wYwrHKOWsmIBmxUgPRrTx9lv36tMlod2xVvWNiTbo=}},
    data: {
        VIN: "1HVBBAANXWH544237",
        LicensePlateNumber: "LS477D",
        State: "WA",
        City: "Tacoma",
        PendingPenaltyTicketAmount: 42.20,
        ValidFromDate: 2011-10-26T,
        ValidToDate: 2023-09-25T,
        Owners: {
            PrimaryOwner: { PersonId: "KmA3XPKKFqYCP2zhR3d0Ho" },
            SecondaryOwners: []
        }
    },
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:0,
        txTime:2019-06-05T20:53:321d-3Z,
        txId:"HgXAkLjAtV0HQ4lNYdzX60"
    }
},
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:17
    },
    hash:{{LGSFZ4iEYWZeMwmAqcxxNyT4wbCtuMOmFCj8pEd6Mp0=}},
    data: {
        VIN: "1HVBBAANXWH544237",
        LicensePlateNumber: "LS477D",
        State: "WA",
        PendingPenaltyTicketAmount: 42.20,
```

```
        ValidFromDate: 2011-10-26T,
        ValidToDate: 2023-09-25T,
        Owners: {
            PrimaryOwner: { PersonId: "KmA3XPKKFqYCP2zhR3d0Ho" },
            SecondaryOwners: []
        },
        City: "Bellevue"
    },
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:1,
        txTime:2019-06-05T21:01:442d-3Z,
        txId:"9cArhIQV5xf5Tf5vtsPwPq"
    }
},
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:19
    },
    hash:{{7bm5DUwpqJFGrmZpb7h9wAxtvggYLPcXq+LAobi9fDg=}},
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:2,
        txTime:2019-06-05T21:03:76d-3Z,
        txId:"9GslbtDtpVHAgYghR5FXbZ"
    }
}
```

Diese Ausgabe enthält Metadatenattribute, die Informationen dazu bieten, wann die einzelnen Elemente geändert wurden und von welcher Transaktion. Aus diesen Daten können Sie Folgendes schließen:

- Das Dokument wird eindeutig durch seine vom System zugewiesene `id` identifiziert: `ADR2Ll1fGsU4Jr4EqTdnQF`. Dies ist eine UUID, die in einer Base62-codierten Zeichenfolge dargestellt wird.
- Eine `INSERT`-Anweisung erstellt die anfängliche Revision eines Dokuments (Version `0`).
- Jedes nachfolgende Update erstellt eine neue Revision mit der gleichen Dokument-`id` und einer aufsteigenden Versionsnummer.
- Das `txId`-Feld gibt die Transaktion an, die jede Revision festgeschrieben hat, und `txTime` zeigt, wann jede Revision festgeschrieben wurde.

- Eine `DELETE`-Anweisung erstellt eine neue, aber finale Revision eines Dokuments. Diese finale Revision hat nur Metadaten.

Um zu erfahren, wie Sie eine Revision dauerhaft löschen, fahren Sie mit fortRedigieren von Dokumentrevisionen.

Redigieren von Dokumentrevisionen

In Amazon QLDB löscht eine`DELETE` Anweisung ein Dokument nur logisch, indem eine neue Revision erstellt wird, die es als gelöscht markiert. QLDB unterstützt auch einen Datenredigierungsvorgang, mit dem Sie inaktive Dokumentrevisionen in der Historie einer Tabelle dauerhaft löschen können.

> **Note**
>
> Alle Bücher, die vor dem 22. Juli 2021 erstellt wurden, können derzeit nicht bearbeitet werden. Sie können die Erstellungszeit Ihres Ledgers in der Amazon QLDB-Konsole einsehen.

Der Redaktionsvorgang löscht nur die Benutzerdaten in der angegebenen Revision und lässt die Journalsequenz und die Dokumentmetadaten unverändert. Dadurch wird die allgemeine Datenintegrität Ihres Ledgers gewahrt.

Bevor Sie mit der Datenredaktion in QLDB beginnen, stellen Sie sicher, dass Sie diesÜberlegungen und Einschränkungen in der Amazon QLDB PartiQL-Referenz überprüfen.

Themen

- Gespeicherte Prozesse
- Überprüfung, ob eine Bearbeitung abgeschlossen ist
- Beispiel für eine Redigierung
- Löschen und Redigieren einer aktiven Revision
- Ein bestimmtes Feld innerhalb einer Revision redigieren

Gespeicherte Prozesse

Sie können dieREVISION REDIGIEREN gespeicherte Prozedur verwenden, um eine einzelne, inaktive Revision in einem Ledger dauerhaft zu löschen. Diese gespeicherte Prozedur löscht alle Benutzerdaten in der angegebenen Revision sowohl im indizierten Speicher als auch im Journalspeicher. Die Journalsequenz und die Metadaten des Dokuments, einschließlich der Dokument-ID und des Hashs, bleiben jedoch unverändert. Dieser Vorgang ist irreversibel.

Bei der angegebenen Dokumentrevision muss es sich um eine inaktive Version in der Historie handeln. Die letzte aktive Revision eines Dokuments kann nicht bearbeitet werden.

Um mehrere Revisionen zu redigieren, müssen Sie die gespeicherte Prozedur für jede Revision einmal ausführen. Sie können eine Revision pro Transaktion redigieren.

Syntax

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

Argumente

`*Blockadresse*`

Der Journalblock, an dem sich die Revision des Dokuments befindet, das redigiert werden soll. Eine Adresse ist eine Amazon Ion-Struktur, die aus zwei Feldern besteht:`strandId` und`sequenceNo`.

Dies ist ein Ion-Literalwert, der mit Backticks bezeichnet wird. Beispiel:

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

'*Tabellen-ID*'

Die eindeutige ID der Tabelle, deren Dokumentversion Sie schwärzen möchten, gekennzeichnet durch einfache Anführungszeichen.

'*Dokument-ID*'

Die eindeutige Dokument-ID der Revision, die redigiert werden soll, gekennzeichnet durch einfache Anführungszeichen.

Überprüfung, ob eine Bearbeitung abgeschlossen ist

Wenn Sie eine Redigierungsanfrage senden, indem Sie die gespeicherte Prozedur ausführen, verarbeitet QLDB die Redigierung von Daten asynchron. Nach Abschluss werden die Benutzerdaten in der Revision (dargestellt durch die`data` Struktur) dauerhaft entfernt. Um zu überprüfen, ob eine Redigierungsanfrage abgeschlossen wurde, können Sie eine der folgenden Methoden verwenden:

- Journalexport
- Journal-Stream
- GetBlock API-Operation
- GetRevision API-Operation
- Verlaufsfunktion— Hinweis: Nachdem eine Schwärzung im Journal abgeschlossen ist, kann es einige Zeit dauern, bis Verlaufsabfragen das Ergebnis der Schwärzung anzeigen. Möglicherweise werden einige Revisionen vor anderen redigiert, wenn die asynchrone Bearbeitung abgeschlossen ist, aber bei Verlaufsabfragen werden irgendwann die abgeschlossenen Ergebnisse angezeigt.

Nach Abschluss einer Revisionsredaktion wird die`data` Struktur der Revision durch ein neues`dataHash` Feld ersetzt. Der Wert dieses Felds ist der Ionen-Hash der entfernten`data` Struktur, wie im folgenden Beispiel gezeigt. Dadurch behält das Ledger seine allgemeine Datenintegrität bei und bleibt durch die vorhandenen Verifizierungs-API-Operationen kryptografisch überprüfbar. Weitere Informationen zur Überprüfung finden Sie unterDatenverifizierung in Amazon QLDB.

Beispiel für eine Redigierung

Beachten Sie den Fahrzeugschein, den Sie zuvor überprüft habenAbfragen des Revisionsverlaufs. Angenommen, Sie wollen die zweite Revision redigieren (`version:1`). Das folgende Abfragebeispiel zeigt diese Revision vor der Bearbeitung. In den Abfrageergebnissen ist die`data` Struktur, die redigiert wird, *`rot kursiv`* hervorgehoben.

```
SELECT * FROM history(VehicleRegistration) AS h
WHERE h.metadata.id = 'ADR2Ll1fGsU4Jr4EqTdnQF' --replace with your id
AND h.metadata.version = 1
```

```
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:17
```

```
    },
    hash:{{LGSFZ4iEYWZeMwmAqcxxNyT4wbCtuMOmFCj8pEd6Mp0=}},
    data: {
        VIN: "1HVBBAANXWH544237",
        LicensePlateNumber: "LS477D",
        State: "WA",
        PendingPenaltyTicketAmount: 42.20,
        ValidFromDate: 2011-10-26T,
        ValidToDate: 2023-09-25T,
        Owners: {
            PrimaryOwner: { PersonId: "KmA3XPKKFqYCP2zhR3d0Ho" },
            SecondaryOwners: []
        },
        City: "Bellevue"
    },
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:1,
        txTime:2019-06-05T21:01:442d-3Z,
        txId:"9cArhIQV5xf5Tf5vtsPwPq"
    }
}
```

Beachten Sie das`blockAddress` in den Abfrageergebnissen, da Sie diesen Wert an die`REDACT_REVISION` gespeicherte Prozedur übergeben müssen. Suchen Sie dann die eindeutige ID der`VehicleRegistration` Tabelle, indem Sie den Systemkatalog wie folgt abfragen.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Verwenden Sie diese Tabellen-ID zusammen mit der Dokument-ID und der Blockadresse, um sie auszuführen`REDACT_REVISION`. Die Tabellen-ID und die Dokument-ID sind Zeichenkettenliterale, die in einfache Anführungszeichen gesetzt werden müssen, und die Blockadresse ist ein Ion-Literal, das in Backticks eingeschlossen ist. Achten Sie darauf, diese Argumente gegebenenfalls durch Ihre eigenen Werte zu ersetzen.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

> **Tip**
>
> Wenn Sie die QLDB-Konsole oder die QLDB-Shell verwenden, um nach einer Tabellen-ID oder Dokument-ID (oder einem beliebigen Zeichenkettenliteralwert) abzufragen, wird der zurückgegebene Wert in doppelte Anführungszeichen gesetzt. Wenn Sie jedoch die Argumente Tabellen-ID und Dokument-ID der`REDACT_REVISION` gespeicherten Prozedur angeben, müssen Sie die Werte in einfache Anführungszeichen setzen.
> Das liegt daran, dass Sie Anweisungen im PartiQL-Format schreiben, QLDB jedoch Ergebnisse im Amazon Ion-Format zurückgibt. Einzelheiten zur Syntax und Semantik von PartiQL in QLDB finden Sie unterAbfragen von Ion mit PartiQL.

Eine gültige Schwärzungsanforderung gibt eine Ion-Struktur zurück, die die Revision des Dokuments darstellt, das Sie redigieren, wie folgt.

```
{
  blockAddress: {
    strandId: "JdxjkR9bSYB5jMHWcI464T",
    sequenceNo: 17
  },
  tableId: "5PLf9SXwndd63lPaSIa0O6",
  documentId: "ADR2Ll1fGsU4Jr4EqTdnQF",
  version: 1
}
```

Wenn Sie diese gespeicherte Prozedur ausführen, verarbeitet QLDB Ihre Redaktionsanforderung asynchron. Nach Abschluss der Bearbeitung wird die`data` Struktur dauerhaft entfernt und durch ein neues *`dataHash`*Feld ersetzt. Der Wert dieses Feldes ist der Ionen-Hash der entfernten`data` Struktur, wie folgt.

> **Note**
>
> Dieses`dataHash` Beispiel dient nur zu Informationszwecken und ist kein wirklich berechneter Hashwert.

```
{
    blockAddress:{
```

```
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:17
    },
    hash:{{LGSFZ4iEYWZeMwmAqcxxNyT4wbCtuMOmFCj8pEd6Mp0=}},
    dataHash: {{s83jd7sfhsdfhksj7hskjdfjfpIPP/DP2hvionas2d4=}},
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:1,
        txTime:2019-06-05T21:01:442d-3Z,
        txId:"9cArhIQV5xf5Tf5vtsPwPq"
    }
}
```

Löschen und Redigieren einer aktiven Revision

Aktive Dokumentrevisionen (d. h. die letzten nicht gelöschten Versionen jedes Dokuments) kommen für die Datenredaktion nicht in Frage. Bevor Sie eine aktive Revision redigieren können, müssen Sie sie zuerst aktualisieren oder löschen. Dadurch wird die zuvor aktive Revision in den Verlauf verschoben und kann bearbeitet werden.

Wenn in Ihrem Anwendungsfall das gesamte Dokument als gelöscht markiert werden muss, verwenden Sie zunächst eine DELETE-Anweisung. Beispielsweise löscht die folgende Anweisung das`VehicleRegistration` Dokument logisch mit einer Fahrgestellnummer von`1HVBBAANXWH544237`.

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

Redigieren Sie dann die vorherige Revision vor diesem Löschen, wie zuvor beschrieben. Bei Bedarf können Sie frühere Änderungen auch individuell redigieren.

Wenn Ihr Anwendungsfall erfordert, dass das Dokument aktiv bleibt, verwenden Sie zunächst eine UPDATE - oder FROM-Anweisung, um die Felder, die Sie schwärzen möchten, zu verdecken oder zu entfernen. Dieser Prozess wird im folgenden Abschnitt beschrieben.

Ein bestimmtes Feld innerhalb einer Revision redigieren

QLDB unterstützt die Bearbeitung eines bestimmten Feldes innerhalb einer Dokumentrevision nicht. Dazu können Sie zunächst eine UPDATE-REMOVE - oder FROM-REMOVE-Anweisung verwenden, um ein vorhandenes Feld aus einer Revision zu entfernen. Mit der folgenden Anweisung

wird beispielsweise das`LicensePlateNumber` Feld mit der Fahrgestellnummer von aus dem`VehicleRegistration` Dokument entfernt`1HVBBAANXWH544237`.

```
UPDATE VehicleRegistration AS r
REMOVE r.LicensePlateNumber
WHERE r.VIN = '1HVBBAANXWH544237'
```

Redigieren Sie dann die vorherige Version vor dieser Entfernung, wie zuvor beschrieben. Bei Bedarf können Sie auch alle früheren Versionen, die dieses jetzt entfernte Feld enthalten, individuell redigieren.

Um zu erfahren, wie Sie Ihre Abfragen optimieren können, fahren Sie mit fortOptimieren der Abfrageleistung.

Optimieren der Abfrageleistung

Amazon QLDB ist darauf ausgelegt, die Anforderungen von Hochleistungs-Workloads bei der Online-Transaktionsverarbeitung (OLTP) zu erfüllen. Das bedeutet, dass QLDB für einen bestimmten Satz von Abfragemustern optimiert ist, obwohl es SQL-ähnliche Abfragefunktionen unterstützt. Es ist wichtig, Anwendungen und ihre Datenmodelle so zu entwerfen, dass sie mit diesen Abfragemustern arbeiten. Andernfalls treten beim Wachstum Ihrer Tabellen erhebliche Leistungsprobleme auf, darunter Abfragelatenz, Transaktions-Timeouts und Parallelitätskonflikte.

In diesem Abschnitt werden Abfragebeschränkungen in QLDB beschrieben und Anleitungen zum Schreiben optimaler Abfragen unter Berücksichtigung dieser Einschränkungen gegeben.

Themen

- Transaktions-Timeout-Limit
- Nebenläufigkeit
- Optimale Abfragemuster
- Abfragemuster, die Sie vermeiden sollten
- Überwachung der Leistung

Transaktions-Timeout-Limit

In QLDB wird jede PartiQL-Anweisung (einschließlich jeder`SELECT` Abfrage) in einer Transaktion verarbeitet und unterliegt einem Transaktions-Timeout-Limit. Eine Transaktion kann bis zu

30 Sekunden lang laufen, bevor sie bestätigt wird. Nach diesem Limit lehnt QLDB alle an der Transaktion geleisteten Arbeiten ab und verwirft die Sitzung, die die Transaktion ausgeführt hat. Dieses Limit schützt den Kunden des Dienstes vor Sitzungsverlusten, indem Transaktionen gestartet werden, ohne sie zu bestätigen oder zu stornieren.

Nebenläufigkeit

QLDB implementiert die Parallelitätssteuerung mithilfe der optimistischen Parallelitätssteuerung (OCC). Suboptimale Abfragen können auch zu mehr OCC-Konflikten führen. Informationen zu OCC finden Sie unterAmazon QLDB-Nebenläufigkeit von Amazon QLDB-Nebenläufigkeit.

Optimale Abfragemuster

Es hat sich bewährt, Anweisungen mit einer`WHERE` Prädikatklausel auszuführen, die nach einem indizierten Feld oder einer Dokument-ID filtert. QLDB benötigt einen Gleichheitsoperator (`=`oder`IN`) für ein indiziertes Feld, um ein Dokument effizient nachschlagen zu können.

Im Folgenden finden Sie Beispiele für optimale Abfragemuster in der Benutzeransicht.

```
--Indexed field (VIN) lookup using the = operator
SELECT * FROM VehicleRegistration
WHERE VIN = '1N4AL11D75C109151'

--Indexed field (VIN) AND non-indexed field (City) lookup
SELECT * FROM VehicleRegistration
WHERE VIN = '1N4AL11D75C109151' AND City = 'Seattle'

--Indexed field (VIN) lookup using the IN operator
SELECT * FROM VehicleRegistration
WHERE VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')

--Document ID (r_id) lookup using the BY clause
SELECT * FROM VehicleRegistration BY r_id
WHERE r_id = '3Qv67yjXEwB9SjmvkuG6Cp'
```

Jede Abfrage, die diesen Mustern nicht folgt, ruft einen vollständigen Tabellenscan auf. Tabellenscans können zu Transaktions-Timeouts für Abfragen in großen Tabellen oder Abfragen, die große Ergebnismengen zurückgeben, führen. Sie können auch zu OCC-Konflikten mit konkurrierenden Transaktionen führen.

Indizes mit hoher Kardinalität

Wir empfehlen, Felder zu indizieren, die Werte mit hoher Kardinalität enthalten. Beispielsweise sind die`LicensePlateNumber` Felder`VIN` und in der`VehicleRegistration` Tabelle indizierte Felder, die eindeutig sein sollen.

Vermeiden Sie es, Felder mit niedriger Kardinalität wie Statuscodes, Adressstaaten oder Provinzen und Postleitzahlen zu indexieren. Wenn Sie ein solches Feld indexieren, können Ihre Abfragen zu großen Ergebnismengen führen, die mit größerer Wahrscheinlichkeit zu Transaktions-Timeouts oder unbeabsichtigten OCC-Konflikten führen.

Engagierte View-Abfragen

Abfragen, die Sie in der Committe-Ansicht ausführen, folgen denselben Optimierungsrichtlinien wie Abfragen in der Benutzeransicht. Indizes, die Sie für eine Tabelle erstellen, werden auch für Abfragen in der Committed-Ansicht verwendet.

Abfragen von Verlaufsfunktionen

Abfragen von Verlaufsfunktionen verwenden nicht die Indizes, die Sie für eine Tabelle erstellen. Der QLDB-Verlauf wird nur nach der Dokument-ID indexiert, und Sie können derzeit keine zusätzlichen Verlaufsindizes erstellen.

Es hat sich bewährt, eine Verlaufsabfrage sowohl mit einem Datumsbereich (Startzeit und Endzeit) als auch mit einer Dokument-ID (`metadata.id`) zu qualifizieren. Verlaufsabfragen, die eine Start- und Endzeit enthalten, profitieren von der Qualifizierung des Datumsbereichs.

Interne Join-Abfragen

Verwenden Sie für interne Join-Abfragen Verknüpfungskriterien, die mindestens ein indiziertes Feld für die Tabelle auf der rechten Seite der Verknüpfung enthalten. Ohne einen Join-Index ruft eine Join-Abfrage mehrere Tabellenscans auf — für jedes Dokument in der linken Tabelle der Verknüpfung scannt die Abfrage die rechte Tabelle vollständig. Die bewährte Methode besteht darin, Felder zu verknüpfen, die für jede Tabelle, die Sie verknüpfen, indexiert sind, und zusätzlich ein`WHERE` Gleichheitsprädikat für mindestens eine Tabelle anzugeben.

Die folgende Abfrage verknüpft beispielsweise die`Vehicle` Tabellen`VehicleRegistration` und in ihren jeweiligen`VIN` Feldern, die beide indexiert sind. Diese Abfrage hat auch ein Gleichheitsprädikat für`VehicleRegistration.VIN`.

```
SELECT * FROM VehicleRegistration AS r INNER JOIN Vehicle AS v
ON r.VIN = v.VIN
```

```
WHERE r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Wählen Sie Indizes mit hoher Kardinalität sowohl für die Verknüpfungskriterien als auch für die Gleichheitsprädikate in Ihren Join-Abfragen.

Abfragemuster, die Sie vermeiden sollten

Im Folgenden finden Sie einige Beispiele für suboptimale Anweisungen, die sich für größere Tabellen in QLDB nicht gut skalieren lassen. Wir empfehlen Ihnen dringend, sich bei Tabellen, die im Laufe der Zeit wachsen, nicht auf diese Art von Abfragen zu verlassen, da Ihre Abfragen letztendlich zu Transaktions-Timeouts führen. Da Tabellen Dokumente mit unterschiedlicher Größe enthalten, ist es schwierig, genaue Grenzwerte für nicht indizierte Abfragen zu definieren.

```
--No predicate clause
SELECT * FROM Vehicle

--COUNT() is not an optimized function
SELECT COUNT(*) FROM Vehicle

--Low-cardinality predicate
SELECT * FROM Vehicle WHERE Color = 'Silver'

--Inequality (>) does not qualify for indexed lookup
SELECT * FROM Vehicle WHERE "Year" > 2019

--Inequality (LIKE)
SELECT * FROM Vehicle WHERE VIN LIKE '1N4AL%'

--Inequality (BETWEEN)
SELECT SUM(PendingPenaltyTicketAmount) FROM VehicleRegistration
WHERE ValidToDate BETWEEN `2020-01-01T` AND `2020-07-01T`

--No predicate clause
DELETE FROM Vehicle

--No document id, and no date range for the history() function
SELECT * FROM history(Vehicle)
```

Im Allgemeinen empfehlen wir nicht, die folgenden Arten von Abfragemustern für produktive Anwendungsfälle in QLDB auszuführen:

- Abfragen zur analytischen Online-Verarbeitung (OLAP)

- Explorative Abfragen ohne Prädikatklausel
- Anfragen melden
- Textsuche

Stattdessen empfehlen wir, Ihre Daten an einen speziell entwickelten Datenbankdienst zu streamen, der für analytische Anwendungsfälle optimiert ist. Sie können beispielsweise QLDB-Daten an Amazon OpenSearch Service streamen, um Volltextsuchfunktionen für Dokumente bereitzustellen. Eine Beispielanwendung, die diesen Anwendungsfall demonstriert, finden Sie im GitHub Repository aws-samples/amazon-qldb-streaming-amazon -opensearch-service-sample-python. Hinweise zu QLDB-Streams finden Sie unterStreaming von Journaldaten aus Amazon QLDB.

Überwachung der Leistung

Der QLDB-Treiber stellt Informationen zur verbrauchten I/O-Nutzung und zum Timing im Ergebnisobjekt einer Anweisung bereit. Sie können diese Metriken verwenden, um ineffiziente PartiQL-Anweisungen zu identifizieren. Um mehr zu erfahren, fahren Sie mit fortAbrufen von PartiQL-Anweisungsstatistiken.

Sie können Amazon auch verwenden CloudWatch , um die Leistung Ihres Ledgers für Datenoperationen zu verfolgen. Überwachen Sie die`CommandLatency` Metrik für ein bestimmtes`LedgerName` und`CommandType`. Weitere Informationen finden Sie unter Überwachung mit Amazon CloudWatch. Informationen dazu, wie QLDB Befehle zur Verwaltung von Datenoperationen verwendet, finden Sie unterSitzungsmanagement mit dem Fahrer.

Abrufen von PartiQL-Anweisungsstatistiken

Amazon QLDB bietet Statistiken zur Anweisungsausführung, die Ihnen helfen können, Ihre Verwendung von QLDB zu optimieren, indem Sie effizientere PartiQL-Anweisungen ausführen. QLDB gibt diese Statistiken zusammen mit den Ergebnissen der Anweisung zurück. Dazu gehören Metriken, die die verbrauchte I/O-Auslastung und die serverseitige Verarbeitungszeit quantifizieren. Anhand dieser Kennzahlen können Sie ineffiziente Aussagen identifizieren.

Diese Funktion ist derzeit im PartiQL-Editor auf der QLDB-Konsole, der QLDB-Shell und der neuesten Version des QLDB-Treibers für alle unterstützten Sprachen verfügbar. In der Konsole können Sie sich auch die Kontoauszugsstatistiken für Ihren Abfrageverlauf anzeigen lassen.

Themen

I/O-Nutzung

Die I/O-Nutzungsmetrik beschreibt die Anzahl der I/O-Lese-Anfragen. Wenn die Anzahl der I/O-Lese-Anfragen höher als erwartet ist, bedeutet dies, dass die Anweisung nicht optimiert ist, z. B. weil kein Index vorhanden ist. Wir empfehlen Ihnen, das vorherige Thema Optimieren der Abfrageleistung zu lesenOptimale Abfragemuster.

> **Note**
>
> Wenn Sie eine`CREATE INDEX` Anweisung für eine nicht leere Tabelle ausführen, enthält die I/O-Nutzungsmetrik nur Leseanforderungen für den Aufruf zur synchronen Indexerstellung. QLDB erstellt den Index für alle vorhandenen Dokumente in der Tabelle asynchron. Diese asynchronen Leseanforderungen sind nicht in der I/O-Nutzungsmetrik Ihrer Abrechnungsergebnisse enthalten. Asynchrone Leseanforderungen werden separat berechnet und nach Abschluss der Indexerstellung zu Ihren gesamten Lese-I/Os hinzugerechnet.

Verwenden der QLDB-Konsole

Gehen Sie wie folgt vor, um die I/O-Nutzung einer Anweisung mithilfe der QLDB-Konsole abzurufen:

1. Öffnen Sie die Amazon QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich PartiQL-Editor aus.
3. Wählen Sie ein Ledger aus der Dropdown-Liste der Ledger aus.
4. Geben Sie im Fenster des Abfrage-Editors eine Anweisung Ihrer Wahl ein, und wählen Sie dann Ausführen. Das Folgende ist ein Abfragebeispiel.

   ```
   SELECT * FROM testTable WHERE firstName = 'Jim'
   ```

   Um eine Anweisung auszuführen, können Sie auch die TastenkombinationCtrl +Enter für Windows oderCmd +Return für macOS verwenden. Weitere Tastenkombinationen finden Sie unter Tastenkombinationen für den PartiQL-Editor.

5. Unter dem Fenster des Abfrage-Editors enthalten Ihre Abfrageergebnisse Lese-I/Os. Dies ist die Anzahl der Leseanforderungen, die von der Anweisung gestellt wurden.

Sie können auch die gelesenen I/Os Ihres Abfrageverlaufs anzeigen, indem Sie die folgenden Schritte ausführen:

1. Wählen Sie im Navigationsbereich unter PartiQL-Editor die Option Letzte Abfragen aus.
2. In der Spalte Read I/Os wird die Anzahl der Leseanforderungen angezeigt, die von jeder Anweisung gestellt wurden.

Verwenden des QLDB-Treibers

Um die I/O-Nutzung einer Anweisung mithilfe des QLDB-Treibers abzurufen, rufen Sie die`getConsumedIOs` Operation des Stream-Cursors oder des gepufferten Cursors des Ergebnisses auf.

In den folgenden Codebeispielen wird gezeigt, wie Sie gelesene I/Os aus dem Stream-Cursor eines Anweisungsergebnisses abrufen.

Java

```
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import software.amazon.qldb.IOUsage;
import software.amazon.qldb.Result;

IonSystem ionSystem = IonSystemBuilder.standard().build();
IonValue ionFirstName = ionSystem.newString("Jim");

driver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM testTable WHERE firstName = ?",
 ionFirstName);

    for (IonValue ionValue : result) {
        // User code here to handle results
    }

    IOUsage ioUsage = result.getConsumedIOs();
    long readIOs = ioUsage.getReadIOs();
```

```
});
```

.NET

```
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

// This is one way of creating Ion values. We can also use a ValueFactory.
// For more details, see: https://docs.aws.amazon.com/qldb/latest/developerguide/
driver-cookbook-dotnet.html#cookbook-dotnet.ion
IIonValue ionFirstName = IonLoader.Default.Load("Jim");

await driver.Execute(async txn =>
{
    IAsyncResult result = await txn.Execute("SELECT * FROM testTable WHERE firstName
 = ?", ionFirstName);

    // Iterate through stream cursor to accumulate read IOs.
    await foreach (IIonValue ionValue in result)
    {
        // User code here to handle results.
        // Warning: It is bad practice to rely on results within a lambda block,
 unless
        // it is to check the state of a result. This is because lambdas are
 retryable.
    }

    var ioUsage = result.GetConsumedIOs();
    var readIOs = ioUsage?.ReadIOs;
});
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

Go

```
import (
```

```
    "context"
    "fmt"
    "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

driver.Execute(context.Background(), func(txn qldbdriver.Transaction) (interface{},
 error) {
    result, err := txn.Execute("SELECT * FROM testTable WHERE firstName = ?", "Jim")

    if err != nil {
        panic(err)
    }

    for result.Next(txn) {
        // User code here to handle results
    }

    ioUsage := result.GetConsumedIOs()
    readIOs := *ioUsage.GetReadIOs()
    fmt.Println(readIOs)
    return nil,nil
})
```

Node.js

```
import { IOUsage, ResultReadable, TransactionExecutor } from "amazon-qldb-driver-
nodejs";

await driver.executeLambda(async (txn: TransactionExecutor) => {
    const result: ResultReadable = await txn.executeAndStreamResults("SELECT * FROM
 testTable WHERE firstName = ?", "Jim");

    for await (const chunk of result) {
        // User code here to handle results
    }

    const ioUsage: IOUsage = result.getConsumedIOs();
    const readIOs: number = ioUsage.getReadIOs();
});
```

Python

```
def get_read_ios(transaction_executor):
```

```
    cursor = transaction_executor.execute_statement("SELECT * FROM testTable WHERE
 firstName = ?", "Jim")

    for row in cursor:
        # User code here to handle results
        pass

    consumed_ios = cursor.get_consumed_ios()
    read_ios = consumed_ios.get('ReadIOs')

qldb_driver.execute_lambda(lambda txn: get_read_ios(txn))
```

In den folgenden Codebeispielen wird gezeigt, wie Sie vom gepufferten Cursor eines Anweisungsergebnisses gelesene I/O abrufen. Dies gibt die Gesamtzahl der gelesenen I/Os von`ExecuteStatement` und`FetchPage` Anfragen zurück.

Java

```
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import software.amazon.qldb.IOUsage;
import software.amazon.qldb.Result;

IonSystem ionSystem = IonSystemBuilder.standard().build();
IonValue ionFirstName = ionSystem.newString("Jim");

Result result = driver.execute(txn -> {
    return txn.execute("SELECT * FROM testTable WHERE firstName = ?", ionFirstName);
});

IOUsage ioUsage = result.getConsumedIOs();
long readIOs = ioUsage.getReadIOs();
```

.NET

```
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

IIonValue ionFirstName = IonLoader.Default.Load("Jim");
```

```
IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM testTable WHERE firstName = ?",
 ionFirstName);
});

var ioUsage = result.GetConsumedIOs();
var readIOs = ioUsage?.ReadIOs;
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

Go

```
import (
    "context"
    "fmt"
    "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    result, err :=  txn.Execute("SELECT * FROM testTable WHERE firstName = ?",
 "Jim")
    if err != nil {
        return nil, err
    }
    return txn.BufferResult(result)
})

if err != nil {
    panic(err)
}

qldbResult := result.(*qldbdriver.BufferedResult)
ioUsage := qldbResult.GetConsumedIOs()
readIOs := *ioUsage.GetReadIOs()
fmt.Println(readIOs)
```

Node.js

```
import { IOUsage, Result, TransactionExecutor } from "amazon-qldb-driver-nodejs";

const result: Result = await driver.executeLambda(async (txn: TransactionExecutor)
 => {
    return await txn.execute("SELECT * FROM testTable WHERE firstName = ?", "Jim");
});

const ioUsage: IOUsage = result.getConsumedIOs();
const readIOs: number = ioUsage.getReadIOs();
```

Python

```
cursor = qldb_driver.execute_lambda(
    lambda txn: txn.execute_statement("SELECT * FROM testTable WHERE firstName = ?",
 "Jim"))

consumed_ios = cursor.get_consumed_ios()
read_ios = consumed_ios.get('ReadIOs')
```

> **Note**
>
> Der Stream-Cursor ist statusmäßig, da er die Ergebnismenge paginiert. Daher geben die`getTimingInformation` Operationen`getConsumedIOs` und die kumulierten Metriken ab dem Zeitpunkt zurück, zu dem Sie sie aufrufen.
> Der gepufferte Cursor puffert die Ergebnismenge im Speicher und gibt die gesamten akkumulierten Metriken zurück.

Informationen zum Zeitpunkt

Die Zeitinformationsmetrik beschreibt die serverseitige Verarbeitungszeit in Millisekunden. Die serverseitige Verarbeitungszeit ist definiert als die Zeit, die QLDB für die Verarbeitung einer Anweisung aufwendet. Dies enthält nicht die für Netzwerkanrufe oder Pausen aufgewendete Zeit. Diese Metrik unterscheidet die Verarbeitungszeit auf der QLDB-Serviceseite von der Verarbeitungszeit auf der Clientseite.

Verwenden der QLDB-Konsole

Gehen Sie wie folgt vor, um die Timing-Informationen einer Anweisung mithilfe der QLDB-Konsole abzurufen:

1. Öffnen Sie die Amazon QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich PartiQL-Editor aus.
3. Wählen Sie ein Ledger aus der Dropdown-Liste der Ledger aus.
4. Geben Sie im Fenster des Abfrage-Editors eine Anweisung Ihrer Wahl ein, und wählen Sie dann Ausführen. Das Folgende ist ein Abfragebeispiel.

   ```
   SELECT * FROM testTable WHERE firstName = 'Jim'
   ```

   Um eine Anweisung auszuführen, können Sie auch die TastenkombinationCtrl +Enter für Windows oderCmd +Return für macOS verwenden. Weitere Tastenkombinationen finden Sie unter Tastenkombinationen für den PartiQL-Editor.
5. Unter dem Fenster des Abfrage-Editors enthalten Ihre Abfrageergebnisse die serverseitige Latenz. Dies ist die Zeitspanne zwischen dem Empfang der Anweisungsanforderung durch QLDB und dem Senden der Antwort. Dies ist ein Subset der gesamten Abfragedauer.

Sie können sich auch die Zeitinformationen Ihres Abfrageverlaufs anzeigen lassen, indem Sie die folgenden Schritte ausführen:

1. Wählen Sie im Navigationsbereich unter PartiQL-Editor die Option Letzte Abfragen aus.
2. In der Spalte Ausführungszeit (ms) werden diese Zeitinformationen für jede Anweisung angezeigt.

Verwenden des QLDB-Treibers

Um die Timing-Informationen einer Anweisung mithilfe des QLDB-Treibers abzurufen, rufen Sie die`getTimingInformation` Operation des Stream-Cursors oder des gepufferten Cursors des Ergebnisses auf.

In den folgenden Codebeispielen wird gezeigt, wie sich die Verarbeitungszeit eines Anweisungsergebnisses aus dem Stream-Cursor holen.

Java

```
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import software.amazon.qldb.Result;
import software.amazon.qldb.TimingInformation;

IonSystem ionSystem = IonSystemBuilder.standard().build();
IonValue ionFirstName = ionSystem.newString("Jim");

driver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM testTable WHERE firstName = ?",
 ionFirstName);

    for (IonValue ionValue : result) {
        // User code here to handle results
    }

    TimingInformation timingInformation = result.getTimingInformation();
    long processingTimeMilliseconds =
 timingInformation.getProcessingTimeMilliseconds();
});
```

.NET

```
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

IIonValue ionFirstName = IonLoader.Default.Load("Jim");

await driver.Execute(async txn =>
{
    IAsyncResult result = await txn.Execute("SELECT * FROM testTable WHERE firstName
 = ?", ionFirstName);

    // Iterate through stream cursor to accumulate processing time.
    await foreach(IIonValue ionValue in result)
    {
        // User code here to handle results.
```

```
        // Warning: It is bad practice to rely on results within a lambda block,
 unless
        // it is to check the state of a result. This is because lambdas are
 retryable.
    }

    var timingInformation = result.GetTimingInformation();
    var processingTimeMilliseconds = timingInformation?.ProcessingTimeMilliseconds;
});
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

Go

```
import (
    "context"
    "fmt"
    "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

driver.Execute(context.Background(), func(txn qldbdriver.Transaction) (interface{},
 error) {
    result, err := txn.Execute("SELECT * FROM testTable WHERE firstName = ?", "Jim")

    if err != nil {
        panic(err)
    }

    for result.Next(txn) {
        // User code here to handle results
    }

    timingInformation := result.GetTimingInformation()
    processingTimeMilliseconds := *timingInformation.GetProcessingTimeMilliseconds()
    fmt.Println(processingTimeMilliseconds)
    return nil, nil
})
```

Node.js

```
import { ResultReadable, TimingInformation, TransactionExecutor } from "amazon-qldb-
driver-nodejs";

await driver.executeLambda(async (txn: TransactionExecutor) => {
    const result: ResultReadable = await txn.executeAndStreamResults("SELECT * FROM
 testTable WHERE firstName = ?", "Jim");

    for await (const chunk of result) {
        // User code here to handle results
    }

    const timingInformation: TimingInformation = result.getTimingInformation();
    const processingTimeMilliseconds: number =
 timingInformation.getProcessingTimeMilliseconds();
});
```

Python

```
def get_processing_time_milliseconds(transaction_executor):
    cursor = transaction_executor.execute_statement("SELECT * FROM testTable WHERE
 firstName = ?", "Jim")

    for row in cursor:
        # User code here to handle results
        pass

    timing_information = cursor.get_timing_information()
    processing_time_milliseconds =
 timing_information.get('ProcessingTimeMilliseconds')

qldb_driver.execute_lambda(lambda txn: get_processing_time_milliseconds(txn))
```

In den folgenden Codebeispielen wird gezeigt, wie Sie die Verarbeitungszeit eines Anweisungsergebnisses aus dem gepufferten Cursor abrufen. Dies gibt die gesamte Verarbeitungszeit von`ExecuteStatement` und`FetchPage` Anfragen zurück.

Java

```
import com.amazon.ion.IonSystem;
```

```
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import software.amazon.qldb.Result;
import software.amazon.qldb.TimingInformation;

IonSystem ionSystem = IonSystemBuilder.standard().build();
IonValue ionFirstName = ionSystem.newString("Jim");

Result result = driver.execute(txn -> {
    return txn.execute("SELECT * FROM testTable WHERE firstName = ?", ionFirstName);
});

TimingInformation timingInformation = result.getTimingInformation();
long processingTimeMilliseconds = timingInformation.getProcessingTimeMilliseconds();
```

.NET

```
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

IIonValue ionFirstName = IonLoader.Default.Load("Jim");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM testTable WHERE firstName = ?",
 ionFirstName);
});

var timingInformation = result.GetTimingInformation();
var processingTimeMilliseconds = timingInformation?.ProcessingTimeMilliseconds;
```

> **Note**
>
> Um in synchronen Code zu konvertieren, entfernen Sie die`async` Schlüsselwörter`await` und und ändern Sie den`IAsyncResult` Typ in`IResult`.

Go

```
import (
```

```
    "context"
    "fmt"
    "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    result, err :=  txn.Execute("SELECT * FROM testTable WHERE firstName = ?",
 "Jim")
    if err != nil {
        return nil, err
    }
    return txn.BufferResult(result)
})

if err != nil {
    panic(err)
}

qldbResult := result.(*qldbdriver.BufferedResult)
timingInformation := qldbResult.GetTimingInformation()
processingTimeMilliseconds := *timingInformation.GetProcessingTimeMilliseconds()
fmt.Println(processingTimeMilliseconds)
```

Node.js

```
import { Result, TimingInformation, TransactionExecutor } from "amazon-qldb-driver-
nodejs";

const result: Result = await driver.executeLambda(async (txn: TransactionExecutor)
 => {
    return await txn.execute("SELECT * FROM testTable WHERE firstName = ?", "Jim");
});

const timingInformation: TimingInformation = result.getTimingInformation();
const processingTimeMilliseconds: number =
 timingInformation.getProcessingTimeMilliseconds();
```

Python

```
cursor = qldb_driver.execute_lambda(
    lambda txn: txn.execute_statement("SELECT * FROM testTable WHERE firstName = ?",
 "Jim"))
```

```
timing_information = cursor.get_timing_information()
processing_time_milliseconds = timing_information.get('ProcessingTimeMilliseconds')
```

> **Note**
>
> Der Stream-Cursor ist statusmäßig, da er die Ergebnismenge paginiert. Daher geben die`getTimingInformation` Operationen`getConsumedIOs` und die kumulierten Metriken ab dem Zeitpunkt zurück, zu dem Sie sie aufrufen.
> Der gepufferte Cursor puffert die Ergebnismenge im Speicher und gibt die gesamten akkumulierten Metriken zurück.

Um zu erfahren, wie Sie den Systemkatalog abfragen, fahren Sie mit Abfragen des Systemkatalogs fort.

Abfragen des Systemkatalogs

Jede Tabelle, die Sie in einem Amazon QLDB-Ledger erstellen, hat eine vom System zugewiesene eindeutige ID. Sie können die ID einer Tabelle, die Liste ihrer Indizes und andere Metadaten ermitteln, indem Sie die Systemkatalogtabelle `information_schema.user_tables` abfragen.

Alle vom System zugewiesenen IDs sind universell eindeutige Identifikatoren (UUID), die jeweils in einer Base62-kodierten Zeichenfolge dargestellt werden. Weitere Informationen finden Sie unter Eindeutige IDs in Amazon QLDB.

Das folgende Beispiel zeigt die Ergebnisse einer Abfrage, die Metadatenattribute der `VehicleRegistration`-Tabelle zurückgibt.

```
SELECT * FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

```
{
    tableId: "5PLf9SXwndd63lPaSIa0O6",
    name: "VehicleRegistration",
    indexes: [
        { indexId: "Djg2nt0yIs2GY0T29Kud1z", expr: "[VIN]", status: "ONLINE" },
        { indexId: "4tPW3fUhaVhDinRgKRLhGU", expr: "[LicensePlateNumber]", status:
 "BUILDING" }
```

```
    ],
    status: "ACTIVE"
}
```

Tabellenmetadatenfelder

- `tableId`— Die eindeutige ID der Tabelle.
- `name`— Der Tabellenname.
- `indexes`— Die Liste der Indizes in der Tabelle.
  - `indexId`— Die eindeutige ID des Index.
  - `expr`— Der indizierte Dokumentpfad. Dieses Feld ist eine Zeichenfolge in der Form:`[fieldName]`.
  - `status`— Der aktuelle Status des Index (`BUILDING`,`FINALIZING`,`ONLINE`,`FAILED`, oder`DELETING`). QLDB verwendet den Index in Abfragen erst, wenn der Status lautet`ONLINE`.
  - `message`— Die Fehlermeldung, die den Grund beschreibt, warum der Index einen`FAILED` Status hat. Dieses Feld ist nur für fehlgeschlagene Indizes enthalten.
- `status`— Der aktuelle Status der Tabelle (`ACTIVE`oder`INACTIVE`). Eine Tabelle wird `INACTIVE`, wenn auf sie `DROP` angewendet wird.

Um zu erfahren, wie Sie Tabellen mit den Anweisungen `DROP TABLE` und `UNDROP TABLE` verwalten, fahren Sie mit Verwalten von Tabellen fort.

Verwalten von Tabellen

Dieser Abschnitt beschreibt, wie Sie Tabellen mithilfe der`DROP TABLE`und`UNDROP TABLE` AND-Anweisungen in Amazon QLDB verwalten. Außerdem wird beschrieben, wie Sie Tabellen taggen, während Sie sie erstellen. Die Kontingente für die Anzahl der aktiven Tabellen und Gesamttabellen, die Sie erstellen können, sind in Kontingente und Limits in Amazon QLDB definiert.

Themen

- Markierung von Tabellen bei der Erstellung
- Tabellen löschen
- Den Verlauf inaktiver Tabellen abfragen
- Tabellen reaktivieren

Markierung von Tabellen bei der Erstellung

> **Note**
>
> Das Markieren von Tabellen bei der Erstellung wird derzeit nur für Ledger im`STANDARD` Berechtigungsmodus unterstützt.

Sie können Ihre Tabellenressourcen markieren. Um Tags für bestehende Tabellen zu verwalten, verwenden Sie dieAWS Management Console oder die API-Operationen`TagResource`,`UntagResource`, und`ListTagsForResource`. Weitere Informationen finden Sie unter Markieren von Amazon-QLDB Ressourcen.

Sie können Tabellentags auch definieren, während Sie die Tabelle erstellen, indem Sie die QLDB-Konsole verwenden oder indem Sie sie in einer`CREATE TABLE` PartiQL-Anweisung angeben. Im folgenden Beispiel wird eine Tabelle erstellt, die`Vehicle` mit dem Tag benannt ist`environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Indem Sie Ressourcen zum Erstellungszeitpunkt markieren, müssen Sie anschließend keine benutzerdefinierten Markierungs-Skripts ausführen. Nachdem eine Tabelle markiert wurde, können Sie den Zugriff auf die Tabelle anhand dieser Tags steuern. Sie können beispielsweise vollen Zugriff nur auf Tabellen gewähren, die ein bestimmtes Tag haben. Ein Beispiel für eine JSON-Richtlinie finden Sie unterVollzugriff auf alle Aktionen basierend auf Tabellen-Tags.

Tabellen löschen

Um eine Tabelle zu entfernen, verwenden Sie eine grundlegende DROP TABLE-Anweisung. Wenn Sie eine Tabelle in QLDB löschen, deaktivieren Sie sie lediglich.

Die folgende Anweisung deaktiviert beispielsweise die`VehicleRegistration` Tabelle.

```
DROP TABLE VehicleRegistration
```

Eine`DROP TABLE` Anweisung gibt die vom System zugewiesene ID der Tabelle zurück. Der Status von`VehicleRegistration` sollte jetzt`INACTIVE` in der Systemkatalogtabelle information_schema.user_tables stehen.

```
SELECT status FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Den Verlauf inaktiver Tabellen abfragen

Zusätzlich zu einem Tabellennamen können Sie die QLDB auchVerlaufsfunktion mit einer Tabellen-ID als erstem Eingabeargument abfragen. Sie müssen die Tabellen-ID verwenden, um den Verlauf einer inaktiven Tabelle abzufragen. Nachdem eine Tabelle deaktiviert wurde, können Sie ihren Verlauf nicht mehr mit dem Tabellennamen abfragen.

Suchen Sie zunächst die Tabellen-ID, indem Sie die Systemkatalogtabelle abfragen. Die folgende Abfrage gibt beispielsweise die `tableId` der `VehicleRegistration`-Tabelle zurück.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Anschließend können Sie diese ID verwenden, um dieselbe Verlaufsabfrage auszuführenAbfragen des Revisionsverlaufs. Im Folgenden finden Sie ein Beispiel, das den Verlauf der Dokument-ID `ADR2Ll1fGsU4Jr4EqTdnQF` aus der Tabellen-ID `5PLf9SXwndd63lPaSIa0O6` abfragt. Die Tabellen-ID ist ein Zeichenfolgenliteralwert, der in einfache Anführungszeichen eingeschlossen werden muss.

```
--replace both the table and document IDs with your values
SELECT * FROM history('5PLf9SXwndd63lPaSIa0O6', `2000T`, `2019-06-05T23:59:59Z`) AS h
WHERE h.metadata.id = 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

Tabellen reaktivieren

Nachdem Sie eine Tabelle in QLDB deaktiviert haben, können Sie dieUNDROP TABLE Anweisung verwenden, um sie zu reaktivieren.

Suchen Sie zuerst die Tabellen-ID von `information_schema.user_tables`. Die folgende Abfrage gibt beispielsweise die `tableId` der `VehicleRegistration`-Tabelle zurück. Der Status sollte `INACTIVE` lauten.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Verwenden Sie dann diese ID, um die Tabelle erneut zu aktivieren. Im Folgenden finden Sie ein Beispiel, das die Tabellen-ID rückgängig macht`5PLf9SXwndd63lPaSIa0O6`. In diesem Fall ist die Tabellen-ID eine eindeutige Kennung, die Sie in doppelte Anführungszeichen setzen.

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

Der Status von `VehicleRegistration` sollte jetzt `ACTIVE` lauten.

Weitere Informationen zum Erstellen, Beschreiben und Löschen von Indizes finden Sie unterVerwalten von Indexen.

Verwalten von Indexen

Dieser Abschnitt beschreibt, wie Sie Indizes in Amazon QLDB erstellen, beschreiben und löschen. Die Kontingente für Anzahl der Indizes pro Tabelle, die Sie erstellen können, ist in definiertKontingente und Limits in Amazon QLDB.

Themen

- Erstellen von Indizes
- Beschreiben von Indizes
- Indizes löschen
- Häufige Fehler

Erstellen von Indizes

Wie auch unter beschriebenErstellen von Tabellen und Indizes, können Sie die CREATE INDEX-Anweisung verwenden, um einen Index für eine Tabelle für ein bestimmtes Feld der obersten Ebene wie folgt zu erstellen. Sowohl der Tabellenname als auch der indizierte Feldname unterscheiden zwischen Groß- und Kleinschreibung.

```
CREATE INDEX ON VehicleRegistration (VIN)
```

```
CREATE INDEX ON VehicleRegistration (LicensePlateNumber)
```

Jeder Index, den Sie für eine Tabelle erstellen, hat eine vom System zugewiesene eindeutige ID. Weitere Informationen zu dieser Index-ID finden Sie im folgenden AbschnittBeschreiben von Indizes.

> **Important**
>
> QLDB benötigt einen Index, um ein Dokument effizient nachschlagen zu können. Ohne Index muss QLDB beim Lesen von Dokumenten einen vollständigen Tabellenscan durchführen. Dies kann bei großen Tabellen zu Leistungsproblemen führen, einschließlich Parallelitätskonflikten und Transaktions-Timeouts.
> Um Tabellenscans zu vermeiden, müssen Sie Anweisungen mit einer`WHERE` Prädikatklausel unter Verwendung eines Gleichheitsoperators (=oder`IN`) für ein indiziertes Feld oder eine Dokument-ID ausführen. Weitere Informationen finden Sie unter Optimieren der Abfrageleistung.

Beachten Sie beim Erstellen von Indizes die folgenden Einschränkungen:

- Ein Index kann nur für ein einzelnes Feld der obersten Ebene erstellt werden. Zusammengesetzte, verschachtelte, eindeutige und funktionsbasierte Indizes werden nicht unterstützt.
- Sie können einen Index für alle Ion-Datentypen erstellen, einschließlich`list` und`struct`. Sie können die indizierte Suche jedoch nur nach Gleichheit des gesamten Ionenwerts durchführen, unabhängig vom Ionentyp. Wenn Sie beispielsweise einen`list` Typ als Index verwenden, können Sie keine indizierte Suche nach einem Element in der Liste durchführen.
- Die Abfrageleistung wird nur verbessert, wenn Sie ein Gleichheitsprädikat verwenden, z. B.`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`.

  QLDB berücksichtigt keine Ungleichungen in Abfrageprädikaten. Daher werden bereichsgefilterte Scans nicht implementiert.
- Bei Namen von indizierten Feldern muss zwischen Groß- und Kleinschreibung unterschieden werden. Sie dürfen höchstens 128 Zeichen lang sein.
- Die Indexerstellung in QLDB erfolgt asynchron. Die Zeit, die zum Erstellen eines Index für eine nicht leere Tabelle benötigt wird, hängt von der Tabellengröße ab. Weitere Informationen finden Sie unter Verwalten von Indexen.

Beschreiben von Indizes

Die Indexerstellung in QLDB erfolgt asynchron. Die Zeit, die zum Erstellen eines Index für eine nicht leere Tabelle benötigt wird, hängt von der Tabellengröße ab. Um den Status einer Indexerstellung zu überprüfen, können Sie die Systemkatalogtabelle information_schema.user_tables abfragen.

Mit der folgenden Anweisung wird beispielsweise der Systemkatalog nach allen Indizes in der`VehicleRegistration` Tabelle abgefragt.

```
SELECT VALUE indexes
FROM information_schema.user_tables info, info.indexes indexes
WHERE info.name = 'VehicleRegistration'
```

```
{
    indexId: "Djg2nt0yIs2GY0T29Kud1z",
    expr: "[VIN]",
    status: "ONLINE"
},
{
    indexId: "4tPW3fUhaVhDinRgKRLhGU",
    expr: "[LicensePlateNumber]",
    status: "FAILED",
    message: "aws.ledger.errors.InvalidEntityError: Document contains multiple values
 for indexed field: LicensePlateNumber"
}
```

Indexfelder

- `indexId`— Die eindeutige ID des Index.
- `expr`— Der indizierte Dokumentpfad. Dieses Feld ist eine Zeichenfolge in der Form:`[fieldName]`.
- `status`— Der aktuelle Status des Index. Der Status eines Index kann einen der folgenden Werte haben:
  - `BUILDING`— Baut aktiv den Index für die Tabelle auf.
  - `FINALIZING`— Hat den Index erstellt und beginnt, ihn für die Verwendung zu aktivieren.
  - `ONLINE`— Ist aktiv und bereit, in Abfragen verwendet zu werden. QLDB verwendet den Index in Abfragen erst, wenn der Status online ist.
  - `FAILED`— Der Index kann aufgrund eines nicht behebbaren Fehlers nicht erstellt werden. Indizes in diesem Status werden weiterhin auf Ihr Indexkontingent pro Tabelle angerechnet. Weitere Informationen finden Sie unter Häufige Fehler.
  - `DELETING`— Löscht aktiv den Index, nachdem ein Benutzer ihn gelöscht hat.
- `message`— Die Fehlermeldung, die den Grund beschreibt, warum der Index einen`FAILED` Status hat. Dieses Feld ist nur für fehlgeschlagene Indizes enthalten.

Verwenden der Konsole

Sie können auch den verwendenAWS Management Console, um den Status eines Index zu überprüfen.

So überprüfen Sie den Status eines Index (-Konsole)

1. Melden Sie sich bei derAWS Management Console an und öffnen Sie die Amazon QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich Ledgers aus.
3. Wählen Sie in der Liste der Bücher den Namen des Buchs aus, dessen Indizes Sie verwalten möchten.
4. Wählen Sie auf der Seite mit den Buchdetails auf der Registerkarte Tabellen den Tabellennamen aus, dessen Index Sie überprüfen möchten.
5. Suchen Sie auf der Seite mit den Tabellendetails die Karte Indizierte Felder. In der Spalte Indexstatus wird der aktuelle Status der einzelnen Indizes in der Tabelle angezeigt.

Indizes löschen

Verwenden Sie dieDROP INDEX Anweisung, um einen Index zu löschen. Wenn Sie einen Index löschen, wird er dauerhaft aus der Tabelle gelöscht.

Suchen Sie zunächst die Index-ID von`information_schema.user_tables`. Die folgende Abfrage gibt beispielsweise den Wert`indexId` des indiziert`LicensePlateNumber` Felds in der`VehicleRegistration` Tabelle zurück.

```
SELECT indexes.indexId
FROM information_schema.user_tables info, info.indexes indexes
WHERE info.name = 'VehicleRegistration' and indexes.expr = '[LicensePlateNumber]'
```

Verwenden Sie dann diese ID, um den Index zu löschen. Im Folgenden finden Sie ein Beispiel, bei dem die Index-ID gelöscht wird`4tPW3fUhaVhDinRgKRLhGU`. Die Index-ID ist ein eindeutiger ID, der in doppelte Anführungszeichen gesetzt werden sollte.

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

> **Note**
>
> Die Klausel`WITH (purge = true)` ist für alle`DROP INDEX` Anweisungen erforderlich und`true` ist derzeit der einzige unterstützte Wert.
> Bei diesem Schlüsselwort muss`purge` es sich um Kleinbuchstaben handeln.

Verwenden der Konsole

Sie können den auch verwendenAWS Management Console, um einen Index zu löschen.

Um einen Index zu löschen (Konsole)

1. Melden Sie sich bei derAWS Management Console an und öffnen Sie die Amazon QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich Ledgers aus.
3. Wählen Sie in der Liste der Bücher den Namen des Buchs aus, dessen Indizes Sie verwalten möchten.
4. Wählen Sie auf der Seite mit den Buchdetails auf der Registerkarte Tabellen den Tabellennamen aus, dessen Index Sie löschen möchten.
5. Suchen Sie auf der Seite mit den Tabellendetails die Karte Indizierte Felder. Wählen Sie den Index aus, den Sie löschen möchten, und wählen Sie dann Index löschen.

Häufige Fehler

In diesem Abschnitt werden häufige Fehler beschrieben, die bei der Erstellung von Indizes auftreten können, und mögliche Lösungen vorgeschlagen.

> **Note**
>
> Indizes mit dem Status von werden`FAILED` immer noch auf Ihr Indexkontingent pro Tabelle angerechnet. Ein fehlerhafter Index verhindert auch, dass Sie Dokumente ändern oder löschen, die dazu geführt haben, dass die Indexerstellung für die Tabelle fehlgeschlagen ist. Sie müssen den Index explizit löschen, um ihn aus dem Kontingent zu entfernen.

Das Dokument enthält mehrere Werte für das indizierte Feld: ***fieldName***.

QLDB kann keinen Index für den angegebenen Feldnamen erstellen, da die Tabelle ein Dokument mit mehreren Werten für dasselbe Feld enthält (d. h. doppelte Feldnamen).

Sie müssen zuerst den fehlerhaften Index löschen. Stellen Sie dann sicher, dass alle Dokumente in der Tabelle nur einen Wert für jeden Feldnamen haben, bevor Sie erneut versuchen, den Index zu erstellen. Sie können auch einen Index für ein anderes Feld erstellen, das keine Duplikate enthält.

QLDB gibt diesen Fehler auch zurück, wenn Sie versuchen, ein Dokument einzufügen, das mehrere Werte für ein Feld enthält, das bereits in der Tabelle indexiert ist.

Indexlimit überschritten: Table ***TableName*** hat bereits ***n*** Indizes und kann keine weiteren erstellen.

QLDB erzwingt ein Limit von fünf Indizes pro Tabelle, einschließlich fehlerhafter Indizes. Sie müssen einen vorhandenen Index löschen, bevor Sie einen neuen erstellen können.

Kein definierter Index mit dem Bezeichner: ***indexID***.

Sie haben versucht, einen Index zu löschen, der für die angegebene Kombination aus Tabelle und Index-ID nicht existiert. Informationen zum Überprüfen vorhandener Indizes finden Sie unterBeschreiben von Indizes.

Eindeutige IDs in Amazon QLDB

In diesem Abschnitt werden die Eigenschaften und Verwendungsrichtlinien von vom System zugewiesenen eindeutigen Identifikatoren in Amazon QLDB beschrieben. Es enthält auch einige Beispiele für eindeutige QLDB-IDs.

Themen

- Eigenschaften
- Verwendung
- Beispiele

Eigenschaften

Alle vom System zugewiesenen IDs in QLDB sind universell eindeutige Identifikatoren (UUID). Jede ID hat die folgenden Eigenschaften:

- 128-Bit-UUID-Nummer
- Dargestellt in Base62-kodiertem Text
- Alphanumerische Zeichenfolge mit fester Länge von 22 Zeichen (zum Beispiel:`3Qv67yjXEwB9SjmvkuG6Cp`)

Verwendung

Beachten Sie bei der Verwendung eindeutiger QLDB-IDs in Ihrer Anwendung die folgenden Richtlinien:

Tun

- Behandeln Sie die ID als Zeichenfolge.

Nicht

- Versuche die Zeichenfolge zu dekodieren.
- Schreiben Sie der Zeichenfolge eine semantische Bedeutung zu (z. B. die Ableitung einer Zeitkomponente).
- Sortiert die Zeichenketten in einer semantischen Reihenfolge.

Beispiele

Die folgenden Attribute sind einige Beispiele für eindeutige IDs in QLDB:

- Dokument-ID
- Index-ID
- Strang-ID
- Tabellen-ID
- Transaktions-ID

Amazon QLDB-Nebenläufigkeit von Amazon QLDB-Nebenläufigkeit

Amazon QLDB ist für Online-Transaktionsverarbeitungs---------- QLDB unterstützt SQL-ähnliche Abfragefunktionen und liefert vollständige ACID-Transaktionen. Darüber hinaus sind QLDB-Datenelemente Dokumente, die Schemaflexibilität und intuitive Datenmodellierung bieten. Mit einem Journal als Herzstück können Sie QLDB verwenden, um auf den vollständigen und überprüfbaren Verlauf aller Änderungen an Ihren Daten zuzugreifen und kohärente Transaktionen bei Bedarf an andere Datendienste zu streamen.

Themen

Optimistic Concurrency Control

In QLDB wird die Parallelitätssteuerung mithilfe der optimistischen Parallelitätssteuerung (OCC) implementiert. OCC arbeitet auf dem Prinzip, dass mehrere Transaktionen häufig abgeschlossen werden können, ohne sich gegenseitig zu beeinflussen.

Mithilfe von OCC werden für Transaktionen in QLDB keine Sperren der Datenbankressourcen übernommen und sie werden mit vollständiger serialisierbarer Isolierung ausgeführt. QLDB führt gleichzeitige Transaktionen seriell aus, sodass sie den gleichen Effekt haben, als ob diese Transaktionen seriell gestartet würden.

Vor dem Commit führt jede Transaktion eine Validierungsprüfung durch, um sicherzustellen, dass keine andere übernommene Transaktion die Daten, auf die sie zugreift, geändert hat. Wenn bei dieser Prüfung widersprüchliche Änderungen festgestellt werden oder sich der Status der Daten ändert, wird die übernehmende Transaktion abgelehnt. Die Transaktion kann jedoch neu gestartet werden.

Wenn eine Transaktion in QLDB schreibt, werden die Validierungsprüfungen des OCC-Modells von QLDB selbst implementiert. Wenn eine Transaktion aufgrund eines Fehlers in der Überprüfungsphase von OCC nicht in das Journal geschrieben werden kann, gibt QLDB`OccConflictException` an die Anwendungsebene zurück. Die Anwendungssoftware ist dafür verantwortlich sicherzustellen, dass die Transaktion neu gestartet wird. Die Anwendung sollte die abgelehnte Transaktion abbrechen und dann die gesamte Transaktion von Anfang an wiederholen.

Informationen dazu, wie der QLDB-Treiber mit OCC-Konflikten und anderen vorübergehenden Ausnahmen umgeht und es erneut versucht, finden Sie unterVerstehen der Wiederholungsrichtlinie mit dem Treiber in Amazon QLDB.

Verwendung von Indizes zur Vermeidung vollständiger Tabellenscans

In QLDB wird jede PartiQL-Anweisung (einschließlich jeder`SELECT` Abfrage) in einer Transaktion verarbeitet und unterliegt einem Transaktions-Timeout-Limit.

Es hat sich bewährt, Anweisungen mit einer`WHERE` Prädikatklausel auszuführen, die nach einem indizierten Feld oder einer Dokument-ID filtert. QLDB benötigt einen Gleichheitsoperator für ein indiziertes Feld, um ein Dokument effizient nachschlagen zu können; zum Beispiel`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`.

Ohne diese indizierte Suche muss QLDB beim Lesen von Dokumenten einen vollständigen Tabellenscan durchführen. Dies kann zu Abfragelatenz und Transaktionszeitüberschreitungen führen und erhöht auch die Wahrscheinlichkeit eines OCC-Konflikts mit konkurrierenden Transaktionen.

Erwägen Sie beispielsweise eine Tabelle mit dem Namen `Vehicle`, die nur einen Index für das Feld `VIN` aufweist. Sie enthält die folgenden Dokumente.

VIN	Marke	Model	Farbe
`"1N4AL11D75C109151"`	`"Audi"`	`"A5"`	`"Silver"`
`"KM8SRDHF6EU074761"`	`"Tesla"`	`"Model S"`	`"Blue"`

VIN	Marke	Model	Farbe
"3HGGK5G5 3FM761765"	"Ducati"	"Monster 1200"	"Yellow"
"1HVBBAAN XWH544237"	"Ford"	"F 150"	"Black"
"1C4RJFAG 0FC625797"	"Mercedes"	"CLK 350"	"White"

Zwei gleichzeitige Benutzer namens Alice und Bob arbeiten mit derselben Tabelle in einem Ledger. Sie möchten zwei verschiedene Dokumente wie folgt aktualisieren.

Alice:

```
UPDATE Vehicle AS v
SET v.Color = 'Blue'
WHERE v.VIN = '1N4AL11D75C109151'
```

Bob:

```
UPDATE Vehicle AS v
SET v.Color = 'Red'
WHERE v.Make = 'Tesla' AND v.Model = 'Model S'
```

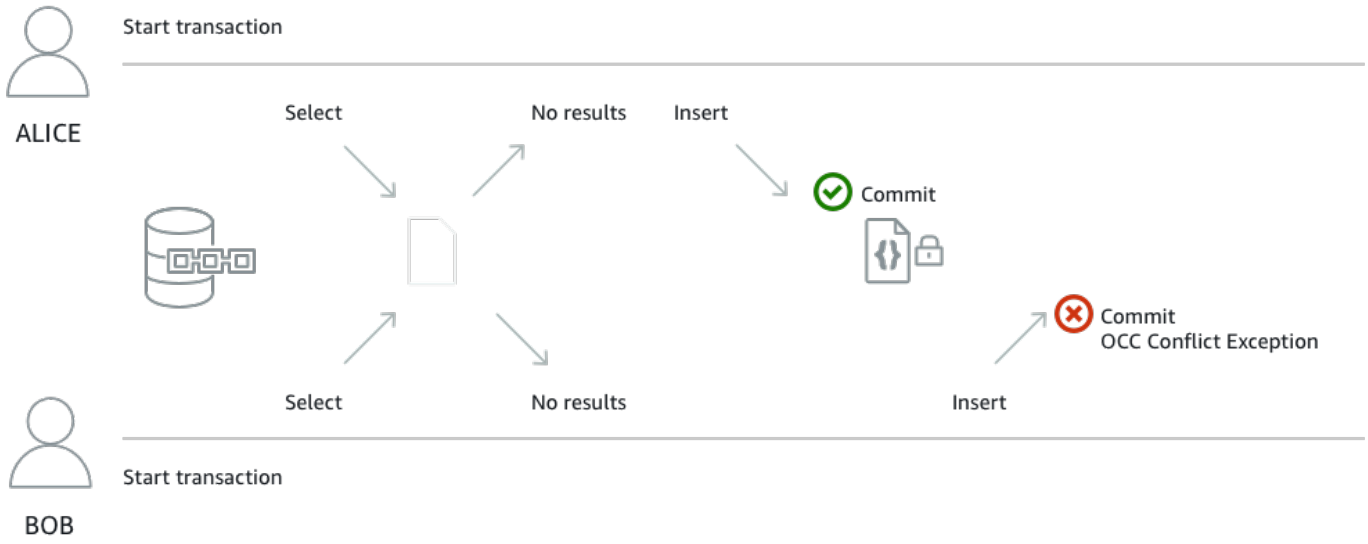
Angenommen, Alice und Bob starten ihre Transaktionen gleichzeitig. Alices `UPDATE`-Anweisung führt eine indizierte Suche für das Feld `VIN` durch, sodass sie nur dieses eine Dokument lesen muss. Alice beendet ihre Transaktion und übergibt diese als erste erfolgreich.

Bobs Anweisung filtert nach nicht indizierten Feldern. Sie führt also einen Tabellenscan durch und trifft auf eine `OccConflictException`. Das liegt daran, dass durch die übernommene Transaktion von Alice die Daten geändert wurden, auf die Bobs Anweisung zugreift. Dazu gehören alle Dokumente in der Tabelle — nicht nur das Dokument, das Bob gerade aktualisiert.

Einfügungs-OCC-Konflikte

OCC-Konflikte können Dokumente beinhalten, die neu eingefügt wurden — nicht nur Dokumente, die zuvor existierten. Sehen Sie sich das folgende Diagramm an, in dem zwei gleichzeitige Benutzer

(Alice und Bob) mit derselben Tabelle in einem Ledger arbeiten. Beide wollen ein neues Dokument nur unter der Bedingung einfügen, dass ein Prädikatwert noch nicht vorhanden ist.



In diesem Beispiel führen sowohl Alice als auch Bob die folgendenSELECTINSERT AND-Anweisungen innerhalb einer einzigen Transaktion aus. Ihre Anwendung führt dieINSERT Anweisung nur aus, wenn dieSELECT Anweisung keine Ergebnisse zurückgibt.

```
SELECT * FROM Vehicle v WHERE v.VIN = 'ABCDE12345EXAMPLE'
```

```
INSERT INTO Vehicle VALUE
{
    'VIN' : 'ABCDE12345EXAMPLE',
    'Type' : 'Wagon',
    'Year' : 2019,
    'Make' : 'Subaru',
    'Model' : 'Outback',
    'Color' : 'Gray'
}
```

Angenommen, Alice und Bob starten ihre Transaktionen gleichzeitig. Beide SELECT Abfragen geben kein vorhandenes Dokument mit einem VIN von ABCDE12345EXAMPLE zurück. Ihre Anwendungen fahren also mit der INSERT-Anweisung fort.

Alice beendet ihre Transaktion und übergibt diese als erste erfolgreich. Dann versucht Bob, seine Transaktion zu bestätigen, aber QLDB lehnt sie ab und wirft eine`OccConflictException`. Dies

liegt daran, dass mit der von Alice übergebenen Transaktion die Ergebnismenge von Bobs `SELECT`-Abfrage geändert wurde und OCC diesen Konflikt erkennt, bevor Bobs Transaktion übergeben wird.

Die`SELECT` Abfrage ist erforderlich, damit dieses Transaktionsbeispiel idempotent ist. Bob kann dann zwar seine gesamte Transaktion von Anfang an wiederholen. Seine nächste`SELECT` Abfrage gibt jedoch das Dokument zurück, das Alice eingefügt hat, sodass Bobs Anwendung das nicht ausführt`INSERT`.

Transaktionen idempotent machen

Die Insert-Transaktion im vorherigen Abschnitt ist auch ein Beispiel für eine idempotente Transaktion. Mit anderen Worten, die mehrfache Ausführung derselben Transaktion führt zu identischen Ergebnissen. Wenn Bob den ausführt,`INSERT` ohne vorher zu überprüfen, ob ein bestimmtes`VIN` Objekt bereits existiert, kann es sein, dass die Tabelle Dokumente enthält, die doppelte`VIN` Werte enthalten.

Berücksichtigen Sie neben OCC-Konflikten auch andere Wiederholungsszenarien. Beispielsweise ist es möglich, dass QLDB eine Transaktion auf der Serverseite erfolgreich festschreibt, der Client jedoch ein Timeout hat, während er auf eine Antwort wartet. Es hat sich bewährt, Schreibtransaktionen idempotent zu machen, um unerwartete Nebenwirkungen im Fall von Parallelität oder Wiederholungsversuchen zu vermeiden.

Redflifliflifliflifliflit

QLDB verhindert das gleichzeitige Schwärzen von Revisionen desselben Journalblocks. Stellen Sie sich ein Beispiel vor, in dem zwei gleichzeitige Benutzer (Alice und Bob) zwei verschiedene Dokumentrevisionen schwärzen möchten, die für denselben Block in einem Hauptbuch eingetragen sind. Zunächst fordert Alice die Bearbeitung einer Revision an, indem sie die`REDACT_REVISION` gespeicherte Prozedur wie folgt ausführt.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

Während Alices Anfrage noch bearbeitet wird, bittet Bob dann wie folgt um die Bearbeitung einer weiteren Revision.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

QLDB lehnt Bobs Anfrage mit einem Kommentar ab,`OccConflictException` obwohl sie versuchen, zwei verschiedene Dokumentrevisionen zu redigieren. Das liegt daran, dass sich Bobs Revision im selben Block befindet wie die Revision, die Alice gerade redigiert. Nachdem die Bearbeitung von Alices Anfrage abgeschlossen ist, kann Bob erneut versuchen, seine Schwärzungsanfrage zu bearbeiten.

Ebenso kann nur eine Anfrage bearbeitet werden, wenn zwei gleichzeitige Transaktionen versuchen, dieselbe Revision zu redigieren. Die andere Anfrage schlägt mit einer OCC-Konfliktexception fehl, bis die Bearbeitung abgeschlossen ist. Danach führen alle Anfragen, dieselbe Revision zu redigieren, zu einem Fehler, der darauf hinweist, dass die Revision bereits redigiert wurde.

Verwalten gleichzeitiger Sitzungen

Wenn Sie Erfahrung mit einem relationalen Datenbankmanagementsystem (RDBMS) haben, sind Sie möglicherweise mit Nebenläufigkeit vertraut. QLDB hat nicht das gleiche Konzept einer herkömmlichen RDBMS-Verbindung, da Transaktionen mit HTTP-Anforderungs- und Antwortnachrichten ausgeführt werden.

In QLDB ist das analoge Konzept eine aktive Sitzung. Eine Sitzung ähnelt konzeptionell einer Benutzeranmeldung — sie verwaltet Informationen über Ihre Datentransaktionsanforderungen an ein Ledger. Eine aktive Sitzung ist eine Sitzung, die aktiv eine Transaktion ausführt. Es kann sich auch um eine Sitzung handeln, die kürzlich eine Transaktion abgeschlossen hat und bei der der Service davon ausgeht, dass er sofort eine weitere Transaktion starten wird. QLDB unterstützt eine aktiv laufende Transaktion pro Sitzung.

Das Limit für gleichzeitige aktive Sitzungen pro Ledger ist in Kontingente und Limits in Amazon QLDB definiert. Nachdem dieses Limit erreicht ist, führt jede Sitzung, die versucht, eine Transaktion zu starten, zu einem Fehler (`LimitExceededException`).

Hinweise zum Lebenszyklus einer Sitzung und zur Verarbeitung von Sitzungen durch den QLDB-Treiber bei der Ausführung von Datentransaktionen finden Sie unterSitzungsmanagement mit dem Fahrer. Bewährte Methoden für die Konfiguration eines Sitzungspool in Ihrer Anwendung mithilfe des QLDB-Treibers finden SieKonfigurieren von QldbDriver Objekt in den Amazon QLDB-Treiberempfehlungen.

Datenverifizierung in Amazon QLDB

Mit Amazon QLDB können Sie darauf vertrauen, dass der Verlauf der Änderungen an Ihren Anwendungsdaten korrekt ist. QLDB verwendet ein unveränderliches Transaktionsprotokoll, das als Journal bezeichnet wird, für die Datenspeicherung. Das Journal verfolgt jede Änderung an Ihren zugesagten Daten und führt einen vollständigen und überprüfbaren Verlauf der Änderungen im Laufe der Zeit.

QLDB verwendet die SHA-256-Hash-Funktion mit einem auf einem Merkle-Baum basierenden Modell, um eine kryptografische Darstellung Ihres Journals zu generieren, die als Digest bezeichnet wird. Der Digest fungiert als eindeutige Signatur des gesamten Änderungsverlaufs Ihrer Daten zu einem bestimmten Zeitpunkt. Sie verwenden den Digest, um die Integrität Ihrer Dokumentrevisionen in Bezug auf diese Signatur zu überprüfen.

Themen

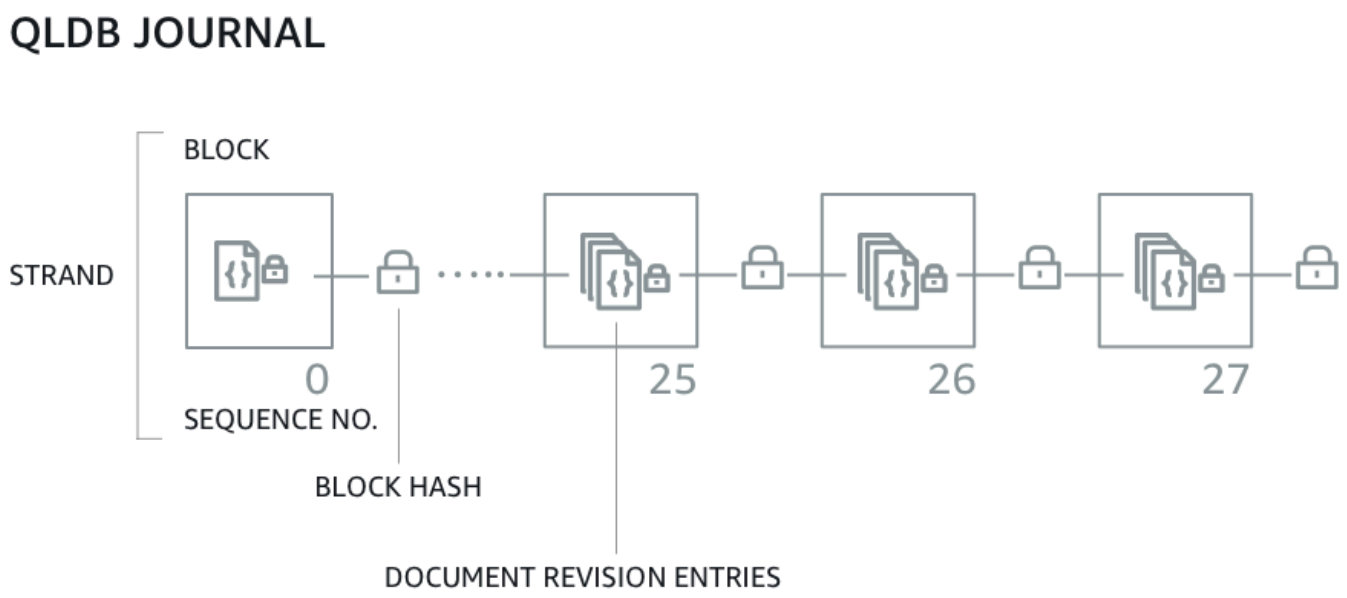
Welche Art von Daten können Sie in QLDB überprüfen?

In QLDB hat jedes Ledger genau ein Journal. Ein Journal kann mehrere Strähnen umfassen. Dabei handelt es sich um Partitionen des Journals.

> **Note**
>
> QLDB unterstützt derzeit nur Journale mit einem einzigen .

Ein Block ist ein Objekt, das bei einer Transaktion in der Journalsträhne festgeschrieben wird. Dieser Block enthält Objekte vom Typ Eintrag, die die Dokumentversionen wiedergeben, die aufgrund der Transaktion entstanden sind. Sie können entweder eine einzelne Revision oder einen gesamten Journalblock in QLDB überprüfen.

Das folgende Diagramm veranschaulicht diese Journalstruktur.



Das Diagramm zeigt, dass Transaktionen in das Journal als Blöcke übergeben werden, die Dokumentrevisionseinträge enthalten. Sie zeigt auch, dass jeder Block mit nachfolgenden Blöcken Hash-verkettet ist und eine Sequenznummer aufweist, um seine Adresse innerhalb der Strähne anzugeben.

Hinweise zum Dateninhalt in einem Block finden Sie unter Journalinhalte in Amazon QLDB.

Was bedeutet Datenintegrität?

Die Datenintegrität in QLDB bedeutet, dass das Journal Ihres Ledgers tatsächlich unveränderlich ist. Mit anderen Worten: Ihre Daten (bzw. alle Dokumentversionen) befinden sich in einem Zustand, in dem die folgenden Bedingungen erfüllt sind:

1. Sie befinden sich an derselben Position im Journal, an der sie ursprünglich geschrieben wurden.
2. Sie wurden seit dem ursprünglichen Schreibvorgang nicht verändert.

Wie funktioniert die Überprüfung?

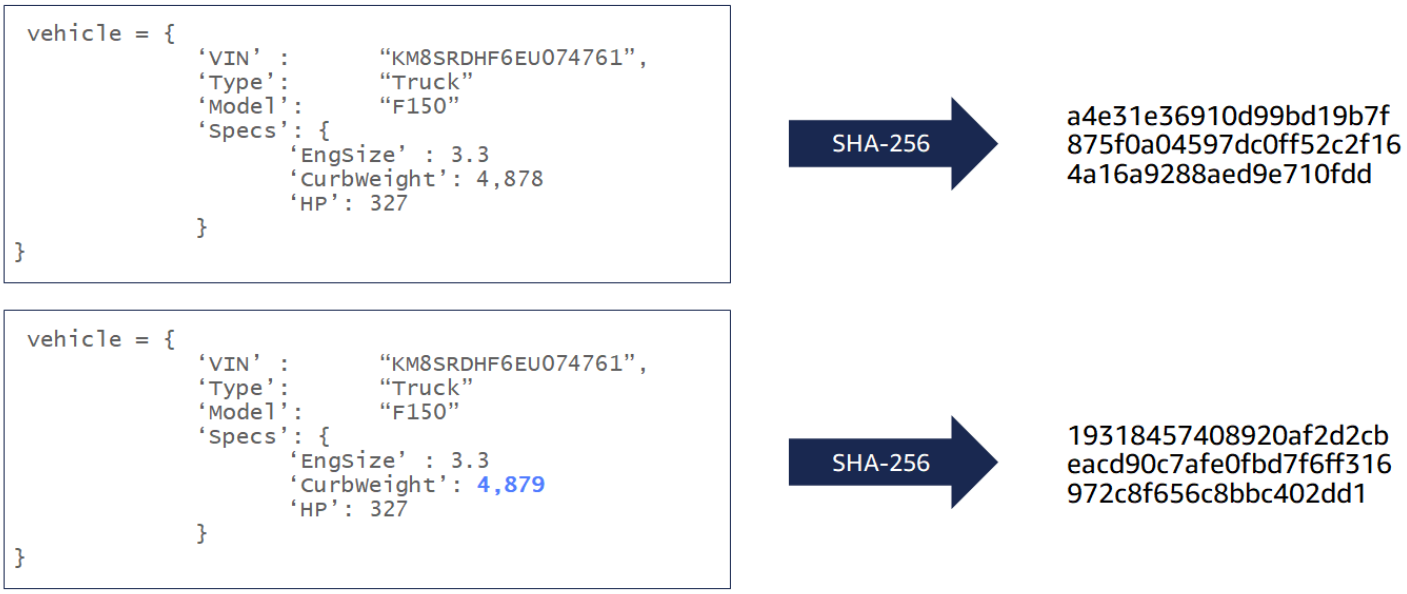
Um zu verstehen, wie die Überprüfung in Amazon QLDB funktioniert, können Sie das Konzept in vier grundlegende Komponenten aufteilen.

- Hashing
- Digest
- Merkle-Baum
- Nachweis

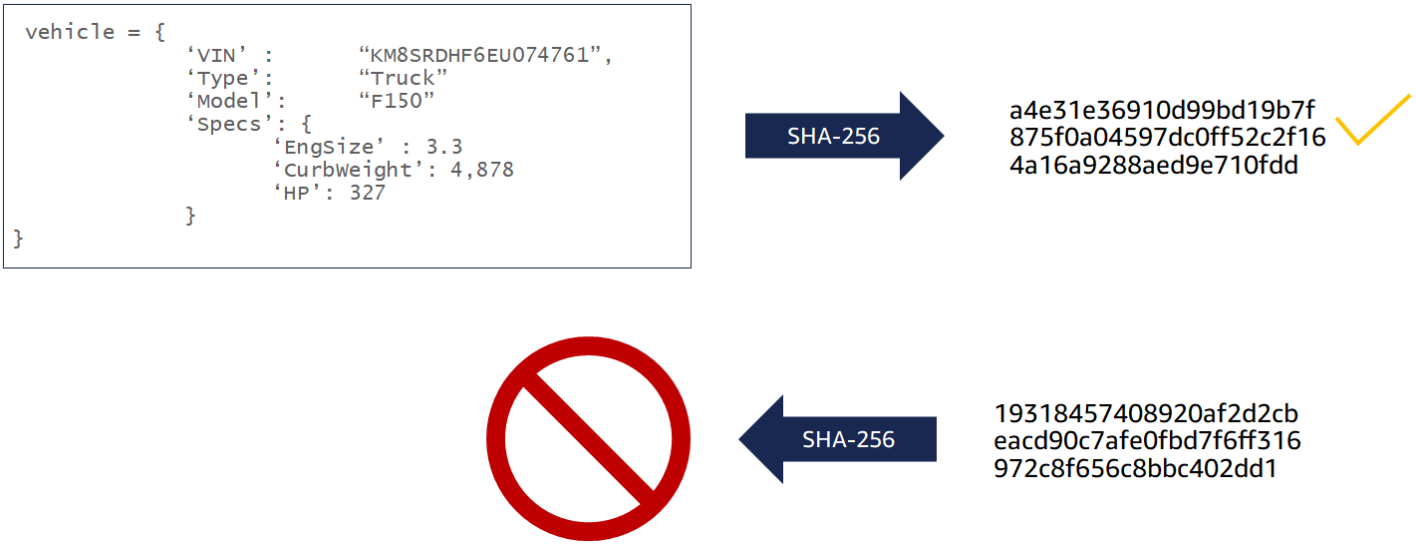
Hashing

QLDB verwendet die kryptografische Hash-Funktion SHA-256, um 256-Bit-Hash-Werte zu erstellen. Ein Hash fungiert als eine eindeutige Signatur mit fester Länge für eine beliebige Anzahl von Eingabedaten. Wenn Sie einen Teil der Eingabe ändern – auch ein einzelnes Zeichen oder Bit – ändert sich der Ausgabe-Hash vollständig.

Das folgende Diagramm zeigt, dass die SHA-256-Hash-Funktion völlig eindeutige Hash-Werte für zwei QLDB-Dokumente erstellt, die sich nur durch eine einzige Ziffer unterscheiden.



Die SHA-256-Hash-Funktion ist eine Möglichkeit, was bedeutet, dass es nicht mathematisch möglich ist, die Eingabe zu berechnen, wenn eine Ausgabe angegeben wird. Das folgende Diagramm zeigt, dass es nicht möglich ist, das Eingabe-QLDB-Dokument zu berechnen, wenn ein Ausgabe-Hash-Wert angegeben wird.



Die folgenden Dateneingaben werden zu Überprüfungszwecken in QLDB gehasht:

- Dokumentversionen
- PartiQL-Anweisungen
- Revisionseinträge
- Journalblöcke

Digest

Ein Digest ist eine kryptografische Darstellung des gesamten Journals Ihres Ledgers zu einem Zeitpunkt. Das Journal kann nur angehängt werden (Append-only) und Journalblöcke werden ähnlich wie Blockchains sequenziert und Hash-verkettet.

Sie können jederzeit einen Digest für einen Ledger anfordern. QLDB generiert den Digest und gibt ihn als sichere Ausgabedatei an Sie zurück. Dann verwenden Sie diesen Digest, um die Integrität von Dokumentrevisionen zu überprüfen, die zu einem früheren Zeitpunkt festgeschrieben wurden. Wenn Sie Hashes neu berechnen, indem Sie mit einer Revision beginnen und mit dem Digest enden, weisen Sie nach, dass Ihre Daten zwischenzeitlich nicht verändert wurden.

Merkle-Baum

Mit zunehmender Größe Ihres Ledgers wird es immer ineffizienter, die vollständige Hash-Kette des Journals zur Überprüfung neu zu berechnen. QLDB verwendet ein Merkle-Baummodell, um diese Ineffizienz zu beheben.

Ein Hash-Baum ist eine Baumdatenstruktur, in der jeder Blattknoten ein Hash eines Datenblocks darstellt. Jeder Nicht-Blattknoten ist ein Hash seiner untergeordneten Knoten. Ein Merkle-Baum wird häufig in Blockchains verwendet und hilft Ihnen dabei, große Datensätze mit einem Prüfnachweismechanismus effizient zu überprüfen. Weitere Informationen zu Hash-Bäumen finden Sie auf der Wikipedia-Seite zu Hash-Bäumen. Weitere Informationen zu Hash-Prüfnachweisen und einen Beispiel-Anwendungsfall finden Sie im Artikel How Log Proofs Work auf der Website von Certificate Transparency.

Die QLDB-Implementierung des Merkle-Baums wird aus der vollständigen Hash-Kette eines Journals erstellt. In diesem Modell sind die Blattknoten die Menge aller einzelnen Dokumentrevisions-Hashes. Der Stammknoten stellt den Digest des gesamten Journals ab einem Zeitpunkt dar.

Mithilfe eines Hash-Baum-Prüfnachweises können Sie eine Revision verifizieren, indem Sie nur eine kleine Teilmenge des Revisionsverlaufs Ihres Ledgers überprüfen. Sie tun dies, indem Sie den Baum von einem bestimmten Blattknoten (Revision) bis zu seiner Stamm (Digest) durchlaufen. Entlang dieses Durchlaufpfads werden Geschwisterpaare von Knoten rekursiv gehasht, um ihren übergeordneten Hash zu berechnen, bis Sie mit dem Digest enden. Dieser Durchlauf weist eine zeitliche Komplexität von `log(n)`-Knoten in der Struktur auf.

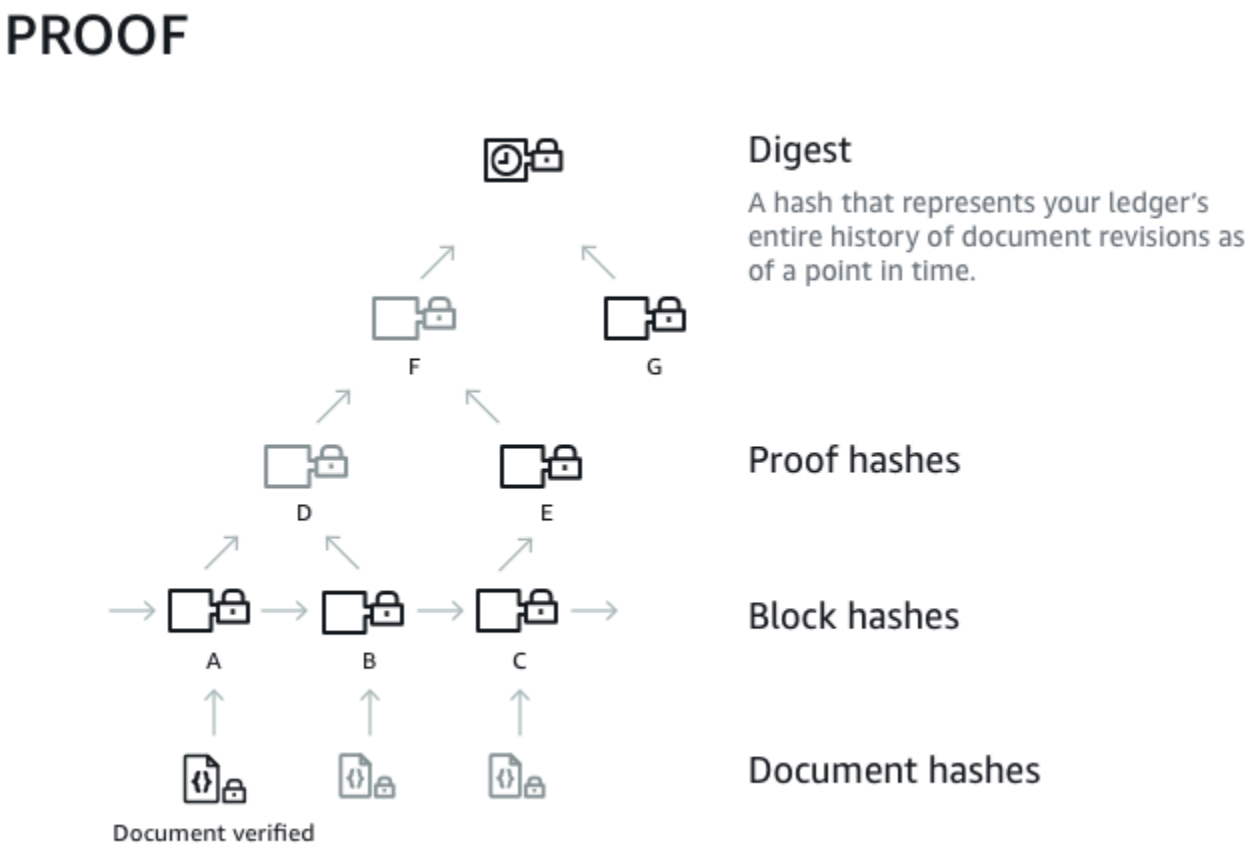
Nachweis

Ein Nachweis ist die geordnete Liste der Knoten-Hashes, die QLDB für einen bestimmten Digest und eine Dokumentrevision zurückgibt. Er besteht aus den Hashes, die von einem Hash-Baummodell benötigt werden, um den gegebenen Blattknoten-Hash (eine Revision) mit dem Stamm-Hash (dem Digest) zu verketten.

Das Ändern von übergebenen Daten zwischen einer Revision und einem Digest unterbricht die Hash-Kette Ihres Journals und macht es unmöglich, einen Nachweis zu generieren.

Verifizierungsbeispiel

Das folgende Diagramm veranschaulicht das Amazon-QLDB-Hash-Baummodell. Es zeigt einen Satz von Block-Hashes, die zum obersten Stammknoten aufgerollt werden, der den Digest einer Journalsträhne darstellt. In einem Ledger mit einem einsträhnigen Journal stellt dieser Stammknoten den Digest des gesamten Ledgers dar.



Angenommen, Knoten A ist der Block, der die Dokumentrevision enthält, deren Hash-Wert Sie verifizieren möchten. Die folgenden Knoten stellen die geordnete Liste der Hashes dar, die QLDB in Ihrem Nachweis bereitstellt: B , E , G . Diese Hashes sind erforderlich, um den Digest aus Hash A neu zu berechnen.

Gehen Sie wie folgt vor, um den Digest neu zu berechnen:

1. Beginnen Sie mit Hash A und verketten Sie es mit Hash B . Hashen Sie dann das Ergebnis, um D zu berechnen.
2. Verwenden Sie D und E, um F zu berechnen.
3. Verwenden Sie F und G, um den Digest zu berechnen.

Die Verifizierung ist erfolgreich, wenn der neu berechnete Digest dem erwarteten Wert entspricht. Bei einem Revisions-Hash und einem Digest ist es nicht möglich, die Hashes in einem Nachweis zurückzuentwickeln. Daher beweist diese Übung, dass Ihre Revision tatsächlich an dieser Stelle des Journals im Verhältnis zum Digest geschrieben wurde.

Wie wirkt sich die Datenredaktion auf die Verifizierung aus?

In Amazon QLDB löscht eine `DELETE` Anweisung ein Dokument nur logisch, indem sie eine neue Revision erstellt, die es als gelöscht markiert. QLDB unterstützt auch eine Datenredaktionsoperation, mit der Sie inaktive Dokumentrevisionen im Verlauf einer Tabelle dauerhaft löschen können.

Die Schwärzungsoperation löscht nur die Benutzerdaten in der angegebenen Revision und lässt die Journalsequenz und die Dokumentmetadaten unverändert. Nachdem eine Revision redigiert wurde, werden die Benutzerdaten in der Revision (dargestellt durch die -`data`Struktur) durch ein neues `dataHash` Feld ersetzt. Der Wert dieses Felds ist der Amazon-Ion-Hash der entfernten `data` Struktur. Weitere Informationen und ein Beispiel für eine Redaktionsoperation finden Sie unter Redigieren von Dokumentrevisionen.

Dadurch behält der Ledger seine allgemeine Datenintegrität bei und bleibt kryptografisch über die vorhandenen Verifizierungs-API-Operationen überprüfbar. Sie können diese API-Operationen weiterhin wie erwartet verwenden, um einen Digest anzufordern (GetDigest), einen Nachweis anzufordern (GetBlock oder GetRevision) und dann Ihren Verifizierungsalgorithmus mit den zurückgegebenen Objekten auszuführen.

Neuberechnung eines Revisions-Hash

Wenn Sie eine einzelne Dokumentrevision überprüfen möchten, indem Sie ihren Hash neu berechnen, müssen Sie bedingt überprüfen, ob die Revision redigiert wurde. Wenn die Revision redigiert wurde, können Sie den im `dataHash` Feld angegebenen Hash-Wert verwenden. Wenn es nicht redigiert wurde, können Sie den Hash mithilfe des `data` Felds neu berechnen.

Durch diese bedingte Prüfung können Sie redigierte Revisionen identifizieren und die entsprechenden Maßnahmen ergreifen. Sie können beispielsweise Datenmanipulationsereignisse zu Überwachungszwecken protokollieren.

Erste Schritte mit der Überprüfung

Bevor Sie Daten überprüfen können, müssen Sie vom Ledger einen Digest anfordern und für später speichern. Jede Dokumentversion, die vor dem neuesten im Digest enthaltenen Block festgeschrieben wurde, ist für die Überprüfung anhand des Digests qualifiziert.

Anschließend fordern Sie einen Nachweis von Amazon QLDB für eine berechtigte Revision an, die Sie überprüfen möchten. Mit diesem Nachweis rufen Sie eine clientseitige API zur Neuberechnung des Digest-Hashes beginnend mit dem Hash Ihrer Version auf. Solange der zuvor gespeicherte Digest außerhalb von QLDB bekannt und vertrauenswürdig ist, wird die Integrität Ihres Dokuments überprüft, wenn Ihr neu berechneter Digest-Hash mit dem gespeicherten Digest-Hash übereinstimmt.

Important

- Was Sie speziell nachweisen, ist, dass die Dokumentrevision zwischen dem Zeitpunkt, an dem Sie diesen Digest gespeichert haben, und dem Zeitpunkt, an dem Sie die Verifizierung ausführen, nicht geändert wurde. Sie können einen Digest anfordern und speichern, sobald eine Revision, die Sie später überprüfen möchten, an das Journal übergeben wurde.
- Als bewährte Methode empfehlen wir Ihnen, regelmäßig Digests anzufordern und sie vom Ledger weg zu speichern. Bestimmen Sie die Häufigkeit, mit der Sie Digests anfordern, basierend darauf, wie oft Sie Revisionen in Ihrem Ledger festschreiben.

  Einen detaillierten AWS Blogbeitrag, der den Wert der kryptografischen Verifizierung im Kontext eines realistischen Anwendungsfalls behandelt, finden Sie unter Reale kryptografische Verifizierung mit Amazon QLDB.

step-by-step Anleitungen zum Anfordern eines Digest von Ihrem Ledger und zum anschließenden Überprüfen Ihrer Daten finden Sie im Folgenden:

Schritt 1: Anfordern eines Digest in QLDB

Amazon QLDB stellt eine API bereit, um einen Digest anzufordern, der den aktuellen Ende des Journals in Ihrem Ledger abdeckt. Der Tipp des Journals bezieht sich auf den letzten festgeschriebenen Block zu dem Zeitpunkt, zu dem QLDB Ihre Anfrage empfängt. Sie können die AWS Management Console, ein AWS SDK oder die AWS Command Line Interface (AWS CLI) verwenden, um einen Digest zu erhalten.

Themen

AWS Management Console

Gehen Sie wie folgt vor, um einen Digest über die QLDB-Konsole anzufordern.

So fordern Sie einen Digest an (Konsole)

1. Melden Sie sich bei der an AWS Management Consoleund öffnen Sie die Amazon-QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich Ledgers aus.
3. Wählen Sie in der Liste der Ledgers den Ledger-Namen, für den Sie einen Digest anfordern möchten.
4. Wählen Sie Get digest (Digest abrufen). Dasm Dialogfeld Get digest (Digest abrufen) zeigt die folgenden Digest-Details:

   - Digest – Der SHA-256-Hashwert des angeforderten Digest.
   - Adresse des Digest-Tipps – Der letzte Blockspeicherort im Journal, der durch den angeforderten Digest abgedeckt wird. Eine Adresse hat die folgenden zwei Felder:
     - `strandId` – Die eindeutige ID des Journaljournals, das den Block enthält.

- `sequenceNo` – Die Indexnummer, die die Position des Blocks innerhalb des angibt.
- Ledger – Der Ledger-Name, für den Sie einen Digest angefordert haben.
- Datum – Der Zeitstempel, zu dem Sie den Digest angefordert haben.

5. Überprüfen Sie die Digest-Informationen. Wählen Sie dann Speichern. Sie können den Standard-Dateinamen behalten oder einen neuen Namen eingeben.

> **Note**
>
> Möglicherweise stellen Sie fest, dass sich Ihre Digest-Hash-Adresswerte ändern, auch wenn Sie keine Daten in Ihrem Ledger ändern. Dies liegt daran, dass die Konsole den Systemkatalog des Ledgers jedes Mal abruft, wenn Sie eine Abfrage im PartiQL-Editor ausführen. Hierbei handelt es sich um eine Read-Transaktion, die an das Journal übertragen wird. Dies führt dazu, dass sich die neueste Block-Adresse ändert.

Dieser Schritt speichert eine Klartext-Datei mit Inhalten im Amazon Ion-Format. Die Datei verfügt über die Dateinamenerweiterung `.ion.txt` und enthält alle Digest-Informationen, die im vorherigen Dialogfeld aufgelistet wurden. Im Folgenden finden Sie ein Beispiel für die Inhalte einer Digest-Datei. Die Reihenfolge der Felder kann je nach Browser variieren.

```
{
  "digest": "42zaJOfV8iGutVGNaIuzQWhD5Xb/5B9lScHnvxPXm9E=",
  "digestTipAddress": "{strandId:\"BlFTjlSXze9BIh1KOszcE3\",sequenceNo:73}",
  "ledger": "my-ledger",
  "date": "2019-04-17T16:57:26.749Z"
}
```

6. Speichern Sie diese Datei an einem Ort, an dem Sie später darauf zugreifen können. Später können Sie diese Datei verwenden, um eine Dokumentrevision zu überprüfen.

> **Important**
>
> Die Dokumentrevision, die Sie zu einem späteren Zeitpunkt überprüfen, muss von dem von Ihnen gespeicherten Digest abgedeckt werden. Das bedeutet, dass die Sequenznummer der Adresse des Dokuments kleiner oder gleich der Sequenznummer der Digest tip address (Digest-Tip-Adresse) sein muss.

QLDB-API

Sie können einen Digest auch von Ihrem Ledger anfordern, indem Sie die Amazon-QLDB-API mit einem - AWS SDK oder der verwenden AWS CLI. Die QLDB-API bietet die folgende Operation zur Verwendung durch Anwendungsprogramme:

- GetDigest – Gibt den Digest eines Ledgers am letzten festgeschriebenen Block im Journal zurück. Die Antwort enthält einen 256-Bit-Hashwert und eine Blockadresse.

Informationen zum Anfordern eines Digest mit der AWS CLIfinden Sie im Befehl get-digest in der AWS CLI -Befehlsreferenz.

Beispielanwendung

Java-Codebeispiele finden Sie im GitHub Repository aws-samples/amazon-qldb-dmv-sample-java . Anweisungen zum Herunterladen und Installieren dieser Beispielanwendung finden Sie unter Installation der Amazon QLDB Java-Beispielanwendung. Bevor Sie einen Digest anfordern, stellen Sie sicher, dass Sie die Schritte 1–3 in der ausführen, Java-Anleitung um einen Beispiel-Ledger zu erstellen und ihn mit Beispieldaten zu laden.

Der Tutorial-Code in der Klasse GetDigest bietet ein Beispiel für die Anforderung eines Digests aus dem Beispiel-Ledger `vehicle-registration`.

Zum Überprüfen einer Dokumentrevision mithilfe des gespeicherten Digests fahren Sie mit Schritt 2: Überprüfen Ihrer Daten in QLDB fort.

Schritt 2: Überprüfen Ihrer Daten in QLDB

Amazon QLDB stellt eine API bereit, um einen Nachweis für eine angegebene Dokument-ID und den zugehörigen Block anzufordern. Sie müssen zudem die Tip-Adresse eines zuvor gespeicherten Digests angeben, wie in Schritt 1: Anfordern eines Digest in QLDB beschrieben. Sie können die AWS Management Console, ein AWS SDK oder die verwenden, AWS CLI um einen Nachweis zu erhalten.

Anschließend können Sie den von QLDB zurückgegebenen Nachweis verwenden, um die Dokumentrevision mithilfe einer clientseitigen API anhand des gespeicherten Digest zu überprüfen. Dadurch haben Sie die Kontrolle über den zum Überprüfen der Daten verwendeten Algorithmus.

Themen

- AWS Management Console
- QLDB-API

AWS Management Console

In diesem Abschnitt werden die Schritte zum Überprüfen einer Dokumentrevision anhand eines zuvor gespeicherten Digest mithilfe der Amazon-QLDB-Konsole beschrieben.

Stellen Sie vor dem Starten sicher, dass Sie die Schritte in Schritt 1: Anfordern eines Digest in QLDB befolgen. Die Überprüfung erfordert einen zuvor gespeicherten Digest, der die Revision abdeckt, die Sie überprüfen möchten.

So überprüfen Sie eine Dokumentrevision (Konsole)

1. Öffnen Sie die Amazon-QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Führen Sie zunächst eine Abfrage Ihres Ledgers für die `id` und die `blockAddress` der Revision durch, die Sie verifizieren möchten. Diese Felder sind in den Metadaten des Dokuments enthalten. Sie können diese Abfrage in der bestätigten Ansicht vornehmen.

   Das Dokument `id` ist eine vom System zugewiesene eindeutige ID-Zeichenfolge. ist `blockAddress` eine Ion-Struktur, die den Blockspeicherort angibt, an dem die Revision festgeschrieben wurde.

   Wählen Sie im Navigationsbereich PartiQL-Editor aus.
3. Wählen Sie den Namen des Ledgers, in dem Sie eine Revision überprüfen möchten.
4. Geben Sie im Abfrage-Editor-Fenster eine `SELECT`-Anweisung in der folgenden Syntax ein und wählen Sie dann Run (Ausführen).

   ```
   SELECT metadata.id, blockAddress FROM _ql_committed_table_name
   WHERE criteria
   ```

   Die folgende Abfrage gibt beispielsweise ein Dokument aus der `VehicleRegistration` Tabelle im Beispiel-Ledger zurück, der in erstellt wurdeErste Schritte mit der Amazon QLDB-Konsole.

   ```
   SELECT r.metadata.id, r.blockAddress FROM _ql_committed_VehicleRegistration AS r
   WHERE r.data.VIN = 'KM8SRDHF6EU074761'
   ```

5. Kopieren und speichern Sie die `id`- und `blockAddress`-Werte, die Ihre Abfrage zurückgibt. Achten Sie darauf, dass Sie die doppelten Anführungszeichen für das `id`-Feld auslassen. In Amazon Ion werden Zeichenfolge-Datentypen in doppelte Anführungszeichen gesetzt. Beispielsweise müssen Sie nur den alphanumerischen Text im folgenden Codeausschnitt kopieren.

   ```
   "LtMNJYNjSwzBLgf7sLifrG"
   ```

6. Nachdem Sie eine Dokumentrevision ausgewählt haben, können Sie damit beginnen, sie zu überprüfen.

   Wählen Sie im Navigationsbereich die Option Verification (Überprüfung) aus.

7. Geben Sie auf dem Formular Verify document (Dokument überprüfen) unter Specify the document that you want to verify (Das Dokument angeben, das Sie überprüfen möchten) die folgenden Eingabeparameter ein:

   - Ledger – Das Ledger, in dem Sie eine Revision überprüfen möchten.
   - Blockadresse – Der von Ihrer Abfrage in Schritt 4 zurückgegebene `blockAddress` Wert.
   - Dokument-ID – Der von Ihrer Abfrage in Schritt 4 zurückgegebene `id` Wert.

8. Wählen Sie unter Specify the document that you want to verify (Das Dokument angeben, das Sie überprüfen möchten) den Digest aus, den Sie zuvor gespeichert haben, indem Sie Choose digest (Digest auswählen) auswählen. Wenn die Datei gültig ist, werden alle Digest-Felder auf Ihrer Konsole automatisch gefüllt. Sie können die folgenden Werte auch manuell direkt aus Ihrer Digest-Datei kopieren und einfügen:

   - Digest – Der `digest` Wert aus Ihrer Digest-Datei.
   - Digest-Tipp-Adresse – Der `digestTipAddress` Wert aus Ihrer Digest-Datei.

9. Überprüfen Sie Ihre Dokument- und Digest-Eingabeparameter und wählen Sie anschließend Verify (Überprüfen) aus.

   Die Konsole automatisiert zwei Schritte für Sie:

   a. Fordern Sie einen Nachweis von QLDB für Ihr angegebenes Dokument an.

   b. Verwenden Sie den von QLDB zurückgegebenen Nachweis, um eine clientseitige API aufzurufen, die Ihre Dokumentrevision anhand des bereitgestellten Digest überprüft. Informationen zum Untersuchen dieses Verifizierungsalgorithmus finden Sie im folgenden Abschnitt, QLDB-API um das Codebeispiel herunterzuladen.

Die Konsole zeigt die Ergebnisse Ihrer Anforderung in der Karte Verification results (Überprüfungsergebnisse) an. Weitere Informationen finden Sie unter Verifizierungsergebnisse.

QLDB-API

Sie können eine Dokumentrevision auch mithilfe der Amazon-QLDB-API mit einem AWS -SDK oder der überprüfen AWS CLI. Die QLDB-API bietet die folgenden Operationen zur Verwendung durch Anwendungsprogramme:

- `GetDigest` – Gibt den Digest eines Ledgers am letzten festgeschriebenen Block im Journal zurück. Die Antwort enthält einen 256-Bit-Hashwert und eine Blockadresse.
- `GetBlock` – Gibt ein Blockobjekt an einer bestimmten Adresse in einem Journal zurück. Gibt auch einen Nachweis für den angegebenen Block zur Überprüfung zurück, wenn `DigestTipAddress` bereitgestellt wird.
- `GetRevision` – Gibt ein Revisionsdatenobjekt für eine angegebene Dokument-ID und Blockadresse zurück. Gibt auch einen Nachweis der angegebenen Revision zur Überprüfung zurück, falls `DigestTipAddress` bereitgestellt wird.

Eine vollständige Beschreibung dieser API-Vorgänge finden Sie unter Amazon QLDB API-Referenz.

Informationen zum Überprüfen von Daten mit der finden Sie in der - AWS CLIBefehlsreferenz. AWS CLI

Beispielanwendung

Java-Codebeispiele finden Sie im GitHub Repository aws-samples/amazon-qldb-dmv-sample-java . Anweisungen zum Herunterladen und Installieren dieser Beispielanwendung finden Sie unter Installation der Amazon QLDB Java-Beispielanwendung. Bevor Sie eine Überprüfung durchführen, stellen Sie sicher, dass Sie die Schritte 1–3 in der ausführen, Java-Anleitung um ein Beispiel-Ledger zu erstellen und es mit Beispieldaten zu laden.

Der Tutorial-Code in der Klasse GetRevision bietet ein Beispiel für das Anfordern eines Nachweises für eine Dokumentrevision und das anschließende Überprüfen dieser Revision. Diese Klasse führt die folgenden Schritte aus:

1. Fordert einen neuen Digest aus dem Beispiel-Ledger `vehicle-registration` an.

2. Fordert einen Nachweis für eine Beispiel-Dokumentrevision aus der Tabelle `VehicleRegistration` im Ledger `vehicle-registration` an.
3. Überprüft die Beispiel-Revision mithilfe des zurückgegebenen Digests und des Nachweises.

Verifizierungsergebnisse

In diesem Abschnitt werden die Ergebnisse beschrieben, die von einer Amazon-QLDB-Datenverifizierungsanforderung auf der zurückgegeben werden AWS Management Console. Detaillierte Anweisungen zum Übermitteln einer Überprüfungsanfrage finden Sie unter Schritt 2: Überprüfen Ihrer Daten in QLDB.

Auf der Seite Verifizierung der QLDB-Konsole werden die Ergebnisse Ihrer Anforderung auf der Karte Verifizierungsergebnisse angezeigt. Auf der Registerkarte Nachweis wird der Inhalt des von QLDB für die angegebene Dokumentrevision und den Digest zurückgegebenen Nachweises angezeigt. Er enthält folgende Details:

- Revisions-Hash – Der SHA-256-Wert, der eindeutig die Dokumentrevision darstellt, die Sie überprüfen.
- Nachweis-Hashes – Die geordnete Liste der von QLDB bereitgestellten Hashes, die zur Neuberechnung des angegebenen Digest verwendet werden. Die Konsole beginnt mit dem Revision hash (Revisions-Hash) und kombiniert ihn der Reihe nach mit jedem Nachweis-Hash, bis sie mit einem neu berechneten Digest endet.

  Die Liste ist standardmäßig ausgeblendet. Sie können sie erweitern, um die Hash-Werte anzuzeigen. Optional können Sie die Hash-Berechnungen selbst testen, indem Sie die in Verwenden eines Beweises zur Neuberechnung Ihres Digest beschriebenen Schritte ausführen.
- Digest berechnet – Der Hash, der sich aus der Reihe von Hash-Berechnungen ergibt, die mit dem Revisions-Hash durchgeführt wurden. Wenn dieser Wert dem zuvor gespeicherten Digest entspricht, ist die Überprüfung erfolgreich.

Auf der Registerkarte Block wird der Inhalt des Blocks angezeigt, der die zu überprüfende Revision enthält. Er enthält folgende Details:

- Transaktions-ID – Die eindeutige ID der Transaktion, die diesen Block bestätigt hat.
- Transaktionszeit – Der Zeitstempel, zu dem dieser Block an den übergeben wurde.
- Block-Hash – Der SHA-256-Wert, der diesen Block und seinen gesamten Inhalt eindeutig darstellt.

- Blockadresse – Der Speicherort im Journal Ihres Ledgers, an dem dieser Block festgeschrieben wurde. Eine Adresse hat die folgenden zwei Felder:
  - BoI-ID – Die eindeutige ID des Journaljournals, das diesen Block enthält.
  - Sequenznummer – Die Indexnummer, die die Position dieses Blocks innerhalb des angibt.
- Anweisungen – Die PartiQL-Anweisungen, die für Commit-Einträge in diesem Block ausgeführt wurden.

> **Note**
>
> Wenn Sie parametrisierte Anweisungen programmgesteuert ausführen, werden sie in Ihren Journalblöcken mit Bindungsparametern anstelle der Literaldaten aufgezeichnet. Sie können beispielsweise die folgende Anweisung in einem Journal-Block anzeigen, wobei das Fragezeichen (?) ein Variablenplatzhalter für den Dokumentinhalt ist.
>
> ```
> INSERT INTO Vehicle ?
> ```

- Dokumenteinträge – Die Dokumentrevisionen, für die in diesem Block ein Commit ausgeführt wurde.

Wenn Ihre Anfrage die Dokumentrevision nicht überprüfen konnte, finden Sie unter Häufige Fehler bei der Überprüfung weitere Informationen zu möglichen Ursachen.

Verwenden eines Beweises zur Neuberechnung Ihres Digest

Nachdem QLDB einen Nachweis für Ihre Dokumentenverifizierungsanforderung zurückgegeben hat, können Sie versuchen, die Hash-Berechnungen selbst durchzuführen. In diesem Abschnitt werden die allgemeinen Schritte zur Neuberechnung Ihres Digest anhand des bereitgestellten Beweises beschrieben.

Koppeln Sie zunächst Ihren Revision hash (Revisions-Hash) mit dem ersten Hash in der Liste Proof hashes (Nachweis-Hashes). Führen Sie dann die folgenden Schritte aus:

1. Sortieren Sie die zwei Hashes. Vergleichen Sie die Hashes anhand ihrer signierten Bytewerte in Little-Endian-Reihenfolge.
2. Verketten Sie die beiden Hashes in sortierter Reihenfolge.
3. Führen Sie für das verkettete Paar einen Hash mit einem SHA-256-Hash-Generator durch.

4. Kombinieren Sie Ihren neuen Hash mit dem nächsten Hash im Nachweis und wiederholen Sie die Schritte 1–3. Nach dem Verarbeiten des letzten Nachweises ist Ihr neuer Hash der neu berechnete Digest.

Wenn Ihr neu berechneter Digest mit dem zuvor gespeicherten Digest übereinstimmt, ist das Dokument erfolgreich überprüft.

Für ein step-by-step Tutorial mit Codebeispielen, die diese Verifizierungsschritte demonstrieren, fahren Sie mit fortTutorial: Überprüfen von Daten mit einem - AWS SDK.

Tutorial: Überprüfen von Daten mit einem - AWS SDK

In diesem Tutorial überprüfen Sie einen Dokumentrevisions-Hash und einen Journalblock-Hash in einem Amazon-QLDB-Ledger mithilfe der QLDB-API über ein - AWS SDK. Sie verwenden auch den QLDB-Treiber, um die Dokumentrevision abzufragen.

Stellen Sie sich ein Beispiel vor, in dem Sie eine Dokumentrevision haben, die Daten für ein Fahrzeug mit einer Fahrzeugidentifikationsnummer (VIN) von enthält`KM8SRDHF6EU074761`. Die Dokumentrevision befindet sich in einer `VehicleRegistration` Tabelle, die sich in einem Ledger mit dem Namen befindet`vehicle-registration`. Angenommen, Sie möchten die Integrität sowohl der Dokumentrevision für dieses Fahrzeug als auch des Journalblocks überprüfen, der die Revision enthält.

> **Note**
>
> Einen ausführlichen AWS Blogbeitrag, in dem der Wert der kryptografischen Verifizierung im Kontext eines realistischen Anwendungsfalls erörtert wird, finden Sie unter Reale kryptografische Verifizierung mit Amazon QLDB.

Themen

Voraussetzungen

Bevor Sie beginnen, stellen Sie sicher, dass die folgende Voraussetzung erfüllt ist:

1. Richten Sie den QLDB-Treiber für eine Sprache Ihrer Wahl ein, indem Sie die entsprechenden Voraussetzungen unter erfüllenErste Schritte mit dem Amazon-QLDB-Treiber. Dazu gehören die Registrierung für AWS, die Gewährung von programmatischem Zugriff für die Entwicklung und die Konfiguration Ihrer Entwicklungsumgebung.
2. Führen Sie die Schritte 1 bis 2 in ausErste Schritte mit der Amazon QLDB-Konsole, um einen Ledger mit dem Namen zu erstellen `vehicle-registration` und ihn mit vordefinierten Beispieldaten zu laden.

Sehen Sie sich als Nächstes die folgenden Schritte an, um zu erfahren, wie die Verifizierung funktioniert, und führen Sie dann das vollständige Codebeispiel von Anfang bis Ende aus.

Schritt 1: Anfordern eines Digest

Bevor Sie Daten überprüfen können, müssen Sie zunächst einen Digest von Ihrem Ledger `vehicle-registration` zur späteren Verwendung anfordern.

Java

```
// Get a digest
GetDigestRequest digestRequest = new GetDigestRequest().withName(ledgerName);
GetDigestResult digestResult = client.getDigest(digestRequest);

java.nio.ByteBuffer digest = digestResult.getDigest();

// expectedDigest is the buffer we will use later to compare against our calculated
 digest
byte[] expectedDigest = new byte[digest.remaining()];
digest.get(expectedDigest);
```

.NET

```
// Get a digest
```

```
GetDigestRequest getDigestRequest = new GetDigestRequest
{
    Name = ledgerName
};
GetDigestResponse getDigestResponse =
 client.GetDigestAsync(getDigestRequest).Result;

// expectedDigest is the buffer we will use later to compare against our calculated
 digest
MemoryStream digest = getDigestResponse.Digest;
byte[] expectedDigest = digest.ToArray();
```

Go

```
// Get a digest
currentLedgerName := ledgerName
input := qldb.GetDigestInput{Name: &currentLedgerName}
digestOutput, err := client.GetDigest(&input)
if err != nil {
    panic(err)
}

// expectedDigest is the buffer we will later use to compare against our calculated
 digest
expectedDigest := digestOutput.Digest
```

Node.js

```
// Get a digest
const getDigestRequest: GetDigestRequest = {
    Name: ledgerName
};
const getDigestResponse: GetDigestResponse = await
 qldbClient.getDigest(getDigestRequest).promise();

// expectedDigest is the buffer we will later use to compare against our calculated
 digest
const expectedDigest: Uint8Array = <Uint8Array>getDigestResponse.Digest;
```

Python

```
# Get a digest
```

```
get_digest_response = qldb_client.get_digest(Name=ledger_name)

# expected_digest is the buffer we will later use to compare against our calculated
 digest
expected_digest = get_digest_response.get('Digest')
digest_tip_address = get_digest_response.get('DigestTipAddress')
```

Schritt 2: Abfragen der Dokumentrevision

Verwenden Sie den QLDB-Treiber, um die Blockadressen, Hashes und Dokument-IDs abzufragen, die der VIN zugeordnet sind`KM8SRDHF6EU074761`.

Java

```
// Retrieve info for the given vin's document revisions
Result result = driver.execute(txn -> {
    final String query = String.format("SELECT blockAddress, hash, metadata.id FROM
 _ql_committed_%s WHERE data.VIN = '%s'", tableName, vin);
    return txn.execute(query);
});
```

.NET

```
// Retrieve info for the given vin's document revisions
var result = driver.Execute(txn => {
    string query = $"SELECT blockAddress, hash, metadata.id FROM
 _ql_committed_{tableName} WHERE data.VIN = '{vin}'";
    return txn.Execute(query);
});
```

Go

```
// Retrieve info for the given vin's document revisions
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    statement := fmt.Sprintf(
            "SELECT blockAddress, hash, metadata.id FROM _ql_committed_%s WHERE
 data.VIN = '%s'",
            tableName,
            vin)
    result, err := txn.Execute(statement)
```

```
    if err != nil {
        return nil, err
    }

    results := make([]map[string]interface{}, 0)

    // Convert the result set into a map
    for result.Next(txn) {
        var doc map[string]interface{}
        err := ion.Unmarshal(result.GetCurrentData(), &doc)
        if err != nil {
            return nil, err
        }
        results = append(results, doc)
    }
    return results, nil
})
if err != nil {
    panic(err)
}
resultSlice := result.([]map[string]interface{})
```

Node.js

```
const result: dom.Value[] = await driver.executeLambda(async (txn:
 TransactionExecutor): Promise<dom.Value[]> => {
    const query: string = `SELECT blockAddress, hash, metadata.id FROM
 _ql_committed_${tableName} WHERE data.VIN = '${vin}'`;
    const queryResult: Result = await txn.execute(query);
    return queryResult.getResultList();
});
```

Python

```
def query_doc_revision(txn):
    query = "SELECT blockAddress, hash, metadata.id FROM _ql_committed_{} WHERE
 data.VIN = '{}'".format(table_name, vin)
    return txn.execute_statement(query)

# Retrieve info for the given vin's document revisions
result = qldb_driver.execute_lambda(query_doc_revision)
```

Schritt 3: Anfordern eines Beweises für die Revision

Führen Sie die Abfrageergebnisse durch und verwenden Sie jede Blockadresse und Dokument-ID zusammen mit dem Ledger-Namen, um eine `GetRevision` Anfrage zu senden. Um einen Nachweis für die Revision zu erhalten, müssen Sie auch die Spitzenadresse aus dem zuvor gespeicherten Digest angeben. Diese API-Operation gibt ein Objekt zurück, das die Dokumentrevision und den Nachweis für die Revision enthält.

Informationen zur Revisionsstruktur und ihren Inhalten finden Sie unter Metadaten von Dokumenten abfragen.

Java

```
for (IonValue ionValue : result) {
    IonStruct ionStruct = (IonStruct)ionValue;

    // Get the requested fields
    IonValue blockAddress = ionStruct.get("blockAddress");
    IonBlob hash = (IonBlob)ionStruct.get("hash");
    String metadataId = ((IonString)ionStruct.get("id")).stringValue();

    System.out.printf("Verifying document revision for id '%s'%n", metadataId);

    String blockAddressText = blockAddress.toString();

    // Submit a request for the revision
    GetRevisionRequest revisionRequest = new GetRevisionRequest()
            .withName(ledgerName)
            .withBlockAddress(new ValueHolder().withIonText(blockAddressText))
            .withDocumentId(metadataId)
            .withDigestTipAddress(digestResult.getDigestTipAddress());

    // Get a result back
    GetRevisionResult revisionResult = client.getRevision(revisionRequest);

    ...
}
```

.NET

```
foreach (IIonValue ionValue in result)
{
```

```
    IIonStruct ionStruct = ionValue;

    // Get the requested fields
    IIonValue blockAddress = ionStruct.GetField("blockAddress");
    IIonBlob hash = ionStruct.GetField("hash");
    String metadataId = ionStruct.GetField("id").StringValue;

    Console.WriteLine($"Verifying document revision for id '{metadataId}'");

    // Use an Ion Reader to convert block address to text
    IIonReader reader = IonReaderBuilder.Build(blockAddress);
    StringWriter sw = new StringWriter();
    IIonWriter textWriter = IonTextWriterBuilder.Build(sw);
    textWriter.WriteValues(reader);
    string blockAddressText = sw.ToString();

    // Submit a request for the revision
    GetRevisionRequest revisionRequest = new GetRevisionRequest
    {
        Name = ledgerName,
        BlockAddress = new ValueHolder
        {
            IonText = blockAddressText
        },
        DocumentId = metadataId,
        DigestTipAddress = getDigestResponse.DigestTipAddress
    };

    // Get a response back
    GetRevisionResponse revisionResponse =
 client.GetRevisionAsync(revisionRequest).Result;

    ...
}
```

Go

```
for _, value := range resultSlice {
    // Get the requested fields
    ionBlockAddress, err := ion.MarshalText(value["blockAddress"])
    if err != nil {
        panic(err)
    }
```

```
    blockAddress := string(ionBlockAddress)
    metadataId := value["id"].(string)
    documentHash := value["hash"].([]byte)

    fmt.Printf("Verifying document revision for id '%s'\n", metadataId)

    // Submit a request for the revision
    revisionInput := qldb.GetRevisionInput{
        BlockAddress:     &qldb.ValueHolder{IonText: &blockAddress},
        DigestTipAddress: digestOutput.DigestTipAddress,
        DocumentId:       &metadataId,
        Name:             &currentLedgerName,
    }

    // Get a result back
    revisionOutput, err := client.GetRevision(&revisionInput)
    if err != nil {
        panic(err)
    }

    ...
}
```

Node.js

```
for (let value of result) {
    // Get the requested fields
    const blockAddress: dom.Value = value.get("blockAddress");
    const hash: dom.Value = value.get("hash");
    const metadataId: string = value.get("id").stringValue();

    console.log(`Verifying document revision for id '${metadataId}'`);

    // Submit a request for the revision
    const revisionRequest: GetRevisionRequest = {
        Name: ledgerName,
        BlockAddress: {
            IonText: dumpText(blockAddress)
        },
        DocumentId: metadataId,
        DigestTipAddress: getDigestResponse.DigestTipAddress
    };
```

```
    // Get a response back
    const revisionResponse: GetRevisionResponse = await
 qldbClient.getRevision(revisionRequest).promise();

    ...
}
```

Python

```
for value in result:
    # Get the requested fields
    block_address = value['blockAddress']
    document_hash = value['hash']
    metadata_id = value['id']

    print("Verifying document revision for id '{}'".format(metadata_id))

    # Submit a request for the revision and get a result back
    proof_response = qldb_client.get_revision(Name=ledger_name,

 BlockAddress=block_address_to_dictionary(block_address),
                                                DocumentId=metadata_id,
                                                DigestTipAddress=digest_tip_address)
```

Rufen Sie dann den Nachweis für die angeforderte Revision ab.

Die QLDB-API gibt den Nachweis als Zeichenfolgendarstellung der geordneten Liste der Knoten-Hashes zurück. Um diese Zeichenfolge in eine Liste der binären Darstellung der Knoten-Hashes zu konvertieren, können Sie einen Ion-Reader aus der Amazon-Ion-Bibliothek verwenden. Weitere Informationen zur Verwendung der Ion-Bibliothek finden Sie im Amazon-Ion-Cookbook .

Java

In diesem Beispiel verwenden Sie , `IonReader` um die binäre Konvertierung durchzuführen.

```
String proofText = revisionResult.getProof().getIonText();

// Take the proof and convert it to a list of byte arrays
List<byte[]> internalHashes = new ArrayList<>();
IonReader reader = SYSTEM.newReader(proofText);
reader.next();
reader.stepIn();
```

```
while (reader.next() != null) {
    internalHashes.add(reader.newBytes());
}
```

.NET

In diesem Beispiel verwenden Sie , `IonLoader` um den Nachweis in ein Ion-Datengram zu laden.

```
string proofText = revisionResponse.Proof.IonText;
IIonDatagram proofValue = IonLoader.Default.Load(proofText);
```

Go

In diesem Beispiel verwenden Sie einen Ion-Reader, um den Nachweis in eine Binärdatei zu konvertieren und die Liste der Knoten-Hashes des Nachweises zu durchlaufen.

```
proofText := revisionOutput.Proof.IonText

// Use ion.Reader to iterate over the proof's node hashes
reader := ion.NewReaderString(*proofText)
// Enter the struct containing node hashes
reader.Next()
if err := reader.StepIn(); err != nil {
    panic(err)
}
```

Node.js

In diesem Beispiel verwenden Sie die -`load`Funktion, um die binäre Konvertierung durchzuführen.

```
let proofValue: dom.Value = load(revisionResponse.Proof.IonText);
```

Python

In diesem Beispiel verwenden Sie die -`loads`Funktion, um die binäre Konvertierung durchzuführen.

```
proof_text = proof_response.get('Proof').get('IonText')
proof_hashes = loads(proof_text)
```

Schritt 4: Den Digest anhand der Revision neu berechnen

Verwenden Sie die Liste der Hashes des Beweises, um den Digest neu zu berechnen, beginnend mit dem Revisions-Hash. Solange der zuvor gespeicherte Digest außerhalb von QLDB bekannt und vertrauenswürdig ist, wird die Integrität der Dokumentrevision überprüft, wenn der neu berechnete Digest-Hash mit dem gespeicherten Digest-Hash übereinstimmt.

Java

```
// Calculate digest
byte[] calculatedDigest = internalHashes.stream().reduce(hash.getBytes(),
 BlockHashVerification::dot);

boolean verified = Arrays.equals(expectedDigest, calculatedDigest);

if (verified) {
    System.out.printf("Successfully verified document revision for id '%s'!%n",
 metadataId);
} else {
    System.out.printf("Document revision for id '%s' verification failed!%n",
 metadataId);
    return;
}
```

.NET

```
byte[] documentHash = hash.Bytes().ToArray();
foreach (IIonValue proofHash in proofValue.GetElementAt(0))
{
    // Calculate the digest
    documentHash = Dot(documentHash, proofHash.Bytes().ToArray());
}

bool verified = expectedDigest.SequenceEqual(documentHash);

if (verified)
{
    Console.WriteLine($"Successfully verified document revision for id
 '{metadataId}'!");
}
else
{
```

```
    Console.WriteLine($"Document revision for id '{metadataId}' verification
 failed!");
    return;
}
```

Go

```
// Going through nodes and calculate digest
for reader.Next() {
    val, _ := reader.ByteValue()
    documentHash, err = dot(documentHash, val)
}

// Compare documentHash with the expected digest
verified := reflect.DeepEqual(documentHash, expectedDigest)

if verified {
    fmt.Printf("Successfully verified document revision for id '%s'!\n", metadataId)
} else {
    fmt.Printf("Document revision for id '%s' verification failed!\n", metadataId)
    return
}
```

Node.js

```
let documentHash: Uint8Array = hash.uInt8ArrayValue();
proofValue.elements().forEach((proofHash: dom.Value) => {
    // Calculate the digest
    documentHash = dot(documentHash, proofHash.uInt8ArrayValue());
});

let verified: boolean = isEqual(expectedDigest, documentHash);

if (verified) {
    console.log(`Successfully verified document revision for id '${metadataId}'!`);
} else {
    console.log(`Document revision for id '${metadataId}' verification failed!`);
    return;
}
```

Python

```
# Calculate digest
```

```
calculated_digest = reduce(dot, proof_hashes, document_hash)

verified = calculated_digest == expected_digest
if verified:
    print("Successfully verified document revision for id
 '{}'!".format(metadata_id))
else:
    print("Document revision for id '{}' verification failed!".format(metadata_id))
```

Schritt 5: Anfordern eines Beweises für den Journalblock

Als Nächstes überprüfen Sie den Journalblock, der die Dokumentrevision enthält.

Verwenden Sie die Blockadresse und die Tipp-Adresse aus dem Digest, den Sie in Schritt 1 gespeichert haben, um eine `GetBlock` Anfrage zu senden. Ähnlich wie bei der `GetRevision` Anforderung in Schritt 2 müssen Sie erneut die Spitzenadresse aus dem gespeicherten Digest angeben, um einen Nachweis für den Block zu erhalten. Diese API-Operation gibt ein Objekt zurück, das den Block und den Nachweis für den Block enthält.

Informationen zur Struktur des Journalblocks und dessen Inhalt finden Sie unter Journalinhalte in Amazon QLDB.

Java

```
// Submit a request for the block
GetBlockRequest getBlockRequest = new GetBlockRequest()
        .withName(ledgerName)
        .withBlockAddress(new ValueHolder().withIonText(blockAddressText))
        .withDigestTipAddress(digestResult.getDigestTipAddress());

// Get a result back
GetBlockResult getBlockResult = client.getBlock(getBlockRequest);
```

.NET

```
// Submit a request for the block
GetBlockRequest getBlockRequest = new GetBlockRequest
{
    Name = ledgerName,
    BlockAddress = new ValueHolder
    {
```

```
        IonText = blockAddressText
    },
    DigestTipAddress = getDigestResponse.DigestTipAddress
};

// Get a response back
GetBlockResponse getBlockResponse = client.GetBlockAsync(getBlockRequest).Result;
```

Go

```
// Submit a request for the block
blockInput := qldb.GetBlockInput{
    Name:             &currentLedgerName,
    BlockAddress:     &qldb.ValueHolder{IonText: &blockAddress},
    DigestTipAddress: digestOutput.DigestTipAddress,
}

// Get a result back
blockOutput, err := client.GetBlock(&blockInput)
if err != nil {
    panic(err)
}
```

Node.js

```
// Submit a request for the block
const getBlockRequest: GetBlockRequest = {
    Name: ledgerName,
    BlockAddress: {
        IonText: dumpText(blockAddress)
    },
    DigestTipAddress: getDigestResponse.DigestTipAddress
};

// Get a response back
const getBlockResponse: GetBlockResponse = await
 qldbClient.getBlock(getBlockRequest).promise();
```

Python

```
def block_address_to_dictionary(ion_dict):
"""
Convert a block address from IonPyDict into a dictionary.
```

```
Shape of the dictionary must be: {'IonText': "{strandId: <"strandId">, sequenceNo:
 <sequenceNo>}"}

:type ion_dict: :py:class:`amazon.ion.simple_types.IonPyDict`/str
:param ion_dict: The block address value to convert.

:rtype: dict
:return: The converted dict.
"""
block_address = {'IonText': {}}
if not isinstance(ion_dict, str):
    py_dict = '{{strandId: "{}", sequenceNo:{}}}'.format(ion_dict['strandId'],
 ion_dict['sequenceNo'])
    ion_dict = py_dict
block_address['IonText'] = ion_dict
return block_address

# Submit a request for the block and get a result back
block_response = qldb_client.get_block(Name=ledger_name,
 BlockAddress=block_address_to_dictionary(block_address),
                                        DigestTipAddress=digest_tip_address)
```

Rufen Sie dann den Block-Hash und den Nachweis aus dem Ergebnis ab.

Java

In diesem Beispiel verwenden Sie , `IonLoader` um das Blockobjekt in einen -`IonDatagram`Container zu laden.

```
String blockText = getBlockResult.getBlock().getIonText();

IonDatagram datagram = SYSTEM.getLoader().load(blockText);
ionStruct = (IonStruct)datagram.get(0);

final byte[] blockHash = ((IonBlob)ionStruct.get("blockHash")).getBytes();
```

Sie verwenden auch `IonLoader`, um den Nachweis in eine zu lade`IonDatagram`n.

```
proofText = getBlockResult.getProof().getIonText();

// Take the proof and create a list of hash binary data
datagram = SYSTEM.getLoader().load(proofText);
```

```
ListIterator<IonValue> listIter =
 ((IonList)datagram.iterator().next()).listIterator();

internalHashes.clear();
while (listIter.hasNext()) {
    internalHashes.add(((IonBlob)listIter.next()).getBytes());
}
```

.NET

In diesem Beispiel verwenden Sie , `IonLoader` um den Block und den Nachweis jeweils in ein Ion-Datengram zu laden.

```
string blockText = getBlockResponse.Block.IonText;
IIonDatagram blockValue = IonLoader.Default.Load(blockText);

// blockValue is a IonDatagram, and the first value is an IonStruct containing the
 blockHash
byte[] blockHash =
 blockValue.GetElementAt(0).GetField("blockHash").Bytes().ToArray();

proofText = getBlockResponse.Proof.IonText;
proofValue = IonLoader.Default.Load(proofText);
```

Go

In diesem Beispiel verwenden Sie einen Ion-Reader, um den Nachweis in eine Binärdatei zu konvertieren und die Liste der Knoten-Hashes des Nachweises zu durchlaufen.

```
proofText = blockOutput.Proof.IonText

block := new(map[string]interface{})
err = ion.UnmarshalString(*blockOutput.Block.IonText, block)
if err != nil {
    panic(err)
}

blockHash := (*block)["blockHash"].([]byte)

// Use ion.Reader to iterate over the proof's node hashes
reader = ion.NewReaderString(*proofText)
// Enter the struct containing node hashes
reader.Next()
```

```
if err := reader.StepIn(); err != nil {
    panic(err)
}
```

Node.js

In diesem Beispiel verwenden Sie die -`load`Funktion, um den -Block und den -Machbarkeitsnachweis in Binärdateien umzuwandeln.

```
const blockValue: dom.Value = load(getBlockResponse.Block.IonText)
let blockHash: Uint8Array = blockValue.get("blockHash").uInt8ArrayValue();

proofValue = load(getBlockResponse.Proof.IonText);
```

Python

In diesem Beispiel verwenden Sie die -`loads`Funktion, um den -Block und den -Machbarkeitsnachweis in Binärdateien umzuwandeln.

```
block_text = block_response.get('Block').get('IonText')
block = loads(block_text)

block_hash = block.get('blockHash')

proof_text = block_response.get('Proof').get('IonText')
proof_hashes = loads(proof_text)
```

Schritt 6: Den Digest aus dem Block neu berechnen

Verwenden Sie die Hash-Liste des Beweises, um den Digest neu zu berechnen, beginnend mit dem Block-Hash. Solange der zuvor gespeicherte Digest außerhalb von QLDB bekannt und vertrauenswürdig ist, wird die Integrität des Blocks überprüft, wenn der neu berechnete Digest-Hash mit dem gespeicherten Digest-Hash übereinstimmt.

Java

```
// Calculate digest
calculatedDigest = internalHashes.stream().reduce(blockHash,
 BlockHashVerification::dot);

verified = Arrays.equals(expectedDigest, calculatedDigest);
```

```
if (verified) {
    System.out.printf("Block address '%s' successfully verified!%n",
 blockAddressText);
} else {
    System.out.printf("Block address '%s' verification failed!%n",
 blockAddressText);
}
```

.NET

```
foreach (IIonValue proofHash in proofValue.GetElementAt(0))
{
    // Calculate the digest
    blockHash = Dot(blockHash, proofHash.Bytes().ToArray());
}

verified = expectedDigest.SequenceEqual(blockHash);

if (verified)
{
    Console.WriteLine($"Block address '{blockAddressText}' successfully verified!");
}
else
{
    Console.WriteLine($"Block address '{blockAddressText}' verification failed!");
}
```

Go

```
// Going through nodes and calculate digest
for reader.Next() {
    val, err := reader.ByteValue()
    if err != nil {
        panic(err)
    }
    blockHash, err = dot(blockHash, val)
}

// Compare blockHash with the expected digest
verified = reflect.DeepEqual(blockHash, expectedDigest)

if verified {
```

```
    fmt.Printf("Block address '%s' successfully verified!\n", blockAddress)
} else {
    fmt.Printf("Block address '%s' verification failed!\n", blockAddress)
    return
}
```

Node.js

```
proofValue.elements().forEach((proofHash: dom.Value) => {
    // Calculate the digest
    blockHash = dot(blockHash, proofHash.uInt8ArrayValue());
});

verified = isEqual(expectedDigest, blockHash);

if (verified) {
    console.log(`Block address '${dumpText(blockAddress)}' successfully verified!`);
} else {
    console.log(`Block address '${dumpText(blockAddress)}' verification failed!`);
}
```

Python

```
# Calculate digest
calculated_digest = reduce(dot, proof_hashes, block_hash)

verified = calculated_digest == expected_digest
if verified:
    print("Block address '{}' successfully verified!".format(dumps(block_address,
                                                                   binary=False,

 omit_version_marker=True)))
else:
    print("Block address '{}' verification failed!".format(block_address))
```

In den vorherigen Codebeispielen wird bei der Neuberechnung des Digest die folgende `dot` Funktion verwendet. Diese Funktion nimmt eine Eingabe von zwei Hashes an, sortiert sie, verkettet sie und gibt dann den Hash des verketteten Arrays zurück.

Java

```
/**
 * Takes two hashes, sorts them, concatenates them, and then returns the
 * hash of the concatenated array.
 *
 * @param h1
 *              Byte array containing one of the hashes to compare.
 * @param h2
 *              Byte array containing one of the hashes to compare.
 * @return the concatenated array of hashes.
 */
public static byte[] dot(final byte[] h1, final byte[] h2) {
    if (h1.length != HASH_LENGTH || h2.length != HASH_LENGTH) {
        throw new IllegalArgumentException("Invalid hash.");
    }

    int byteEqual = 0;
    for (int i = h1.length - 1; i >= 0; i--) {
        byteEqual = Byte.compare(h1[i], h2[i]);
        if (byteEqual != 0) {
            break;
        }
    }

    byte[] concatenated = new byte[h1.length + h2.length];
    if (byteEqual < 0) {
        System.arraycopy(h1, 0, concatenated, 0, h1.length);
        System.arraycopy(h2, 0, concatenated, h1.length, h2.length);
    } else {
        System.arraycopy(h2, 0, concatenated, 0, h2.length);
        System.arraycopy(h1, 0, concatenated, h2.length, h1.length);
    }

    MessageDigest messageDigest;
    try {
        messageDigest = MessageDigest.getInstance("SHA-256");
    } catch (NoSuchAlgorithmException e) {
        throw new IllegalStateException("SHA-256 message digest is unavailable", e);
    }

    messageDigest.update(concatenated);
    return messageDigest.digest();
```

```
}
```

.NET

```
/// <summary>
/// Takes two hashes, sorts them, concatenates them, and then returns the
/// hash of the concatenated array.
/// </summary>
/// <param name="h1">Byte array containing one of the hashes to compare.</param>
/// <param name="h2">Byte array containing one of the hashes to compare.</param>
/// <returns>The concatenated array of hashes.</returns>
private static byte[] Dot(byte[] h1, byte[] h2)
{
    if (h1.Length == 0)
    {
        return h2;
    }

    if (h2.Length == 0)
    {
        return h1;
    }

    HashAlgorithm hashAlgorithm = HashAlgorithm.Create("SHA256");
    HashComparer comparer = new HashComparer();
    if (comparer.Compare(h1, h2) < 0)
    {
        return hashAlgorithm.ComputeHash(h1.Concat(h2).ToArray());
    }
    else
    {
        return hashAlgorithm.ComputeHash(h2.Concat(h1).ToArray());
    }
}

private class HashComparer : IComparer<byte[]>
{
    private static readonly int HASH_LENGTH = 32;

    public int Compare(byte[] h1, byte[] h2)
    {
        if (h1.Length != HASH_LENGTH || h2.Length != HASH_LENGTH)
        {
```

```
            throw new ArgumentException("Invalid hash");
        }

        for (var i = h1.Length - 1; i >= 0; i--)
        {
            var byteEqual = (sbyte)h1[i] - (sbyte)h2[i];
            if (byteEqual != 0)
            {
                return byteEqual;
            }
        }

        return 0;
    }
}
```

Go

```
// Takes two hashes, sorts them, concatenates them, and then returns the hash of the
 concatenated array.
func dot(h1, h2 []byte) ([]byte, error) {
    compare, err := hashComparator(h1, h2)
    if err != nil {
        return nil, err
    }

    var concatenated []byte
    if compare < 0 {
        concatenated = append(h1, h2...)
    } else {
        concatenated = append(h2, h1...)
    }

    newHash := sha256.Sum256(concatenated)
    return newHash[:], nil
}

func hashComparator(h1 []byte, h2 []byte) (int16, error) {
    if len(h1) != hashLength || len(h2) != hashLength {
        return 0, errors.New("invalid hash")
    }
    for i := range h1 {
        // Reverse index for little endianness
```

```
        index := hashLength - 1 - i

        // Handle byte being unsigned and overflow
        h1Int := int16(h1[index])
        h2Int := int16(h2[index])
        if h1Int > 127 {
            h1Int = 0 - (256 - h1Int)
        }
        if h2Int > 127 {
            h2Int = 0 - (256 - h2Int)
        }

        difference := h1Int - h2Int
        if difference != 0 {
            return difference, nil
        }
    }
    return 0, nil
}
```

Node.js

```
/**
 * Takes two hashes, sorts them, concatenates them, and calculates a digest based on
 the concatenated hash.
 * @param h1 Byte array containing one of the hashes to compare.
 * @param h2 Byte array containing one of the hashes to compare.
 * @returns The digest calculated from the concatenated hash values.
 */
function dot(h1: Uint8Array, h2: Uint8Array): Uint8Array {
    if (h1.length === 0) {
        return h2;
    }
    if (h2.length === 0) {
        return h1;
    }

    const newHashLib = createHash("sha256");

    let concatenated: Uint8Array;
    if (hashComparator(h1, h2) < 0) {
        concatenated = concatenate(h1, h2);
    } else {
```

```
        concatenated = concatenate(h2, h1);
    }
    newHashLib.update(concatenated);
    return newHashLib.digest();
}

/**
 * Compares two hashes by their **signed** byte values in little-endian order.
 * @param hash1 The hash value to compare.
 * @param hash2 The hash value to compare.
 * @returns Zero if the hash values are equal, otherwise return the difference of
 the first pair of non-matching
 *          bytes.
 * @throws RangeError When the hash is not the correct hash size.
 */
function hashComparator(hash1: Uint8Array, hash2: Uint8Array): number {
    if (hash1.length !== HASH_SIZE || hash2.length !== HASH_SIZE) {
        throw new RangeError("Invalid hash.");
    }
    for (let i = hash1.length-1; i >= 0; i--) {
        const difference: number = (hash1[i]<<24 >>24) - (hash2[i]<<24 >>24);
        if (difference !== 0) {
            return difference;
        }
    }
    return 0;
}

/**
 * Helper method that concatenates two Uint8Array.
 * @param arrays List of arrays to concatenate, in the order provided.
 * @returns The concatenated array.
 */
function concatenate(...arrays: Uint8Array[]): Uint8Array {
    let totalLength = 0;
    for (const arr of arrays) {
        totalLength += arr.length;
    }
    const result = new Uint8Array(totalLength);
    let offset = 0;
    for (const arr of arrays) {
        result.set(arr, offset);
        offset += arr.length;
    }
```

```
    return result;
}

/**
 * Helper method that checks for equality between two Uint8Array.
 * @param expected Byte array containing one of the hashes to compare.
 * @param actual Byte array containing one of the hashes to compare.
 * @returns Boolean indicating equality between the two Uint8Array.
 */
function isEqual(expected: Uint8Array, actual: Uint8Array): boolean {
    if (expected === actual) return true;
    if (expected == null || actual == null) return false;
    if (expected.length !== actual.length) return false;

    for (let i = 0; i < expected.length; i++) {
        if (expected[i] !== actual[i]) {
            return false;
        }
    }
    return true;
}
```

Python

```
def dot(hash1, hash2):
    """
    Takes two hashes, sorts them, concatenates them, and then returns the
    hash of the concatenated array.

    :type hash1: bytes
    :param hash1: The hash value to compare.

    :type hash2: bytes
    :param hash2: The hash value to compare.

    :rtype: bytes
    :return: The new hash value generated from concatenated hash values.
    """
    if len(hash1) != hash_length or len(hash2) != hash_length:
        raise ValueError('Illegal hash.')

    hash_array1 = array('b', hash1)
    hash_array2 = array('b', hash2)
```

```
    difference = 0
    for i in range(len(hash_array1) - 1, -1, -1):
        difference = hash_array1[i] - hash_array2[i]
        if difference != 0:
            break

    if difference < 0:
        concatenated = hash1 + hash2
    else:
        concatenated = hash2 + hash1

    new_hash_lib = sha256()
    new_hash_lib.update(concatenated)
    new_digest = new_hash_lib.digest()
    return new_digest
```

Ausführen des vollständigen Codebeispiels

Führen Sie das vollständige Codebeispiel wie folgt aus, um alle vorherigen Schritte von Anfang bis Ende auszuführen.

Java

```
import com.amazon.ion.IonBlob;
import com.amazon.ion.IonDatagram;
import com.amazon.ion.IonList;
import com.amazon.ion.IonReader;
import com.amazon.ion.IonString;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.AmazonQLDBClientBuilder;
import com.amazonaws.services.qldb.model.GetBlockRequest;
import com.amazonaws.services.qldb.model.GetBlockResult;
import com.amazonaws.services.qldb.model.GetDigestRequest;
import com.amazonaws.services.qldb.model.GetDigestResult;
import com.amazonaws.services.qldb.model.GetRevisionRequest;
import com.amazonaws.services.qldb.model.GetRevisionResult;
import com.amazonaws.services.qldb.model.ValueHolder;
```

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.ListIterator;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.qldbsession.QldbSessionClient;
import software.amazon.awssdk.services.qldbsession.QldbSessionClientBuilder;
import software.amazon.qldb.QldbDriver;
import software.amazon.qldb.Result;

public class BlockHashVerification {
    private static final IonSystem SYSTEM = IonSystemBuilder.standard().build();
    private static final QldbDriver driver = createQldbDriver();
    private static final AmazonQLDB client =
 AmazonQLDBClientBuilder.standard().build();
    private static final String region = "us-east-1";
    private static final String ledgerName = "vehicle-registration";
    private static final String tableName = "VehicleRegistration";
    private static final String vin = "KM8SRDHF6EU074761";
    private static final int HASH_LENGTH = 32;

    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static QldbDriver createQldbDriver() {
        QldbSessionClientBuilder sessionClientBuilder = QldbSessionClient.builder();
        sessionClientBuilder.region(Region.of(region));

        return QldbDriver.builder()
                .ledger(ledgerName)
                .sessionClientBuilder(sessionClientBuilder)
                .build();
    }

    /**
     * Takes two hashes, sorts them, concatenates them, and then returns the
     * hash of the concatenated array.
     *
     * @param h1
```

```
     *              Byte array containing one of the hashes to compare.
     * @param h2
     *              Byte array containing one of the hashes to compare.
     * @return the concatenated array of hashes.
     */
    public static byte[] dot(final byte[] h1, final byte[] h2) {
        if (h1.length != HASH_LENGTH || h2.length != HASH_LENGTH) {
            throw new IllegalArgumentException("Invalid hash.");
        }

        int byteEqual = 0;
        for (int i = h1.length - 1; i >= 0; i--) {
            byteEqual = Byte.compare(h1[i], h2[i]);
            if (byteEqual != 0) {
                break;
            }
        }

        byte[] concatenated = new byte[h1.length + h2.length];
        if (byteEqual < 0) {
            System.arraycopy(h1, 0, concatenated, 0, h1.length);
            System.arraycopy(h2, 0, concatenated, h1.length, h2.length);
        } else {
            System.arraycopy(h2, 0, concatenated, 0, h2.length);
            System.arraycopy(h1, 0, concatenated, h2.length, h1.length);
        }

        MessageDigest messageDigest;
        try {
            messageDigest = MessageDigest.getInstance("SHA-256");
        } catch (NoSuchAlgorithmException e) {
            throw new IllegalStateException("SHA-256 message digest is unavailable",
e);
        }

        messageDigest.update(concatenated);
        return messageDigest.digest();
    }

    public static void main(String[] args) {
        // Get a digest
        GetDigestRequest digestRequest = new
GetDigestRequest().withName(ledgerName);
        GetDigestResult digestResult = client.getDigest(digestRequest);
```

```
        java.nio.ByteBuffer digest = digestResult.getDigest();

        // expectedDigest is the buffer we will use later to compare against our
calculated digest
        byte[] expectedDigest = new byte[digest.remaining()];
        digest.get(expectedDigest);

        // Retrieve info for the given vin's document revisions
        Result result = driver.execute(txn -> {
            final String query = String.format("SELECT blockAddress, hash,
metadata.id FROM _ql_committed_%s WHERE data.VIN = '%s'", tableName, vin);
            return txn.execute(query);
        });

        System.out.printf("Verifying document revisions for vin '%s' in table '%s'
in ledger '%s'%n", vin, tableName, ledgerName);

        for (IonValue ionValue : result) {
            IonStruct ionStruct = (IonStruct)ionValue;

            // Get the requested fields
            IonValue blockAddress = ionStruct.get("blockAddress");
            IonBlob hash = (IonBlob)ionStruct.get("hash");
            String metadataId = ((IonString)ionStruct.get("id")).stringValue();

            System.out.printf("Verifying document revision for id '%s'%n",
metadataId);

            String blockAddressText = blockAddress.toString();

            // Submit a request for the revision
            GetRevisionRequest revisionRequest = new GetRevisionRequest()
                    .withName(ledgerName)
                    .withBlockAddress(new
ValueHolder().withIonText(blockAddressText))
                    .withDocumentId(metadataId)
                    .withDigestTipAddress(digestResult.getDigestTipAddress());

            // Get a result back
            GetRevisionResult revisionResult = client.getRevision(revisionRequest);

            String proofText = revisionResult.getProof().getIonText();
```

```
            // Take the proof and convert it to a list of byte arrays
            List<byte[]> internalHashes = new ArrayList<>();
            IonReader reader = SYSTEM.newReader(proofText);
            reader.next();
            reader.stepIn();
            while (reader.next() != null) {
                internalHashes.add(reader.newBytes());
            }

            // Calculate digest
            byte[] calculatedDigest =
internalHashes.stream().reduce(hash.getBytes(), BlockHashVerification::dot);

            boolean verified = Arrays.equals(expectedDigest, calculatedDigest);

            if (verified) {
                System.out.printf("Successfully verified document revision for id
'%s'!%n", metadataId);
            } else {
                System.out.printf("Document revision for id '%s' verification
failed!%n", metadataId);
                return;
            }

            // Submit a request for the block
            GetBlockRequest getBlockRequest = new GetBlockRequest()
                    .withName(ledgerName)
                    .withBlockAddress(new
ValueHolder().withIonText(blockAddressText))
                    .withDigestTipAddress(digestResult.getDigestTipAddress());

            // Get a result back
            GetBlockResult getBlockResult = client.getBlock(getBlockRequest);

            String blockText = getBlockResult.getBlock().getIonText();

            IonDatagram datagram = SYSTEM.getLoader().load(blockText);
            ionStruct = (IonStruct)datagram.get(0);

            final byte[] blockHash =
((IonBlob)ionStruct.get("blockHash")).getBytes();

            proofText = getBlockResult.getProof().getIonText();
```

```
            // Take the proof and create a list of hash binary data
            datagram = SYSTEM.getLoader().load(proofText);
            ListIterator<IonValue> listIter =
 ((IonList)datagram.iterator().next()).listIterator();

            internalHashes.clear();
            while (listIter.hasNext()) {
                internalHashes.add(((IonBlob)listIter.next()).getBytes());
            }

            // Calculate digest
            calculatedDigest = internalHashes.stream().reduce(blockHash,
 BlockHashVerification::dot);

            verified = Arrays.equals(expectedDigest, calculatedDigest);

            if (verified) {
                System.out.printf("Block address '%s' successfully verified!%n",
 blockAddressText);
            } else {
                System.out.printf("Block address '%s' verification failed!%n",
 blockAddressText);
            }
        }
    }
}
```

.NET

```
using Amazon.IonDotnet;
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
using Amazon.QLDB;
using Amazon.QLDB.Driver;
using Amazon.QLDB.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Cryptography;

namespace BlockHashVerification
{
```

```
    class BlockHashVerification
    {
        private static readonly string ledgerName = "vehicle-registration";
        private static readonly string tableName = "VehicleRegistration";
        private static readonly string vin = "KM8SRDHF6EU074761";
        private static readonly IQldbDriver driver =
 QldbDriver.Builder().WithLedger(ledgerName).Build();
        private static readonly IAmazonQLDB client = new AmazonQLDBClient();

        /// <summary>
        /// Takes two hashes, sorts them, concatenates them, and then returns the
        /// hash of the concatenated array.
        /// </summary>
        /// <param name="h1">Byte array containing one of the hashes to compare.</
param>
        /// <param name="h2">Byte array containing one of the hashes to compare.</
param>
        /// <returns>The concatenated array of hashes.</returns>
        private static byte[] Dot(byte[] h1, byte[] h2)
        {
            if (h1.Length == 0)
            {
                return h2;
            }

            if (h2.Length == 0)
            {
                return h1;
            }

            HashAlgorithm hashAlgorithm = HashAlgorithm.Create("SHA256");
            HashComparer comparer = new HashComparer();
            if (comparer.Compare(h1, h2) < 0)
            {
                return hashAlgorithm.ComputeHash(h1.Concat(h2).ToArray());
            }
            else
            {
                return hashAlgorithm.ComputeHash(h2.Concat(h1).ToArray());
            }
        }

        private class HashComparer : IComparer<byte[]>
        {
```

```
            private static readonly int HASH_LENGTH = 32;

            public int Compare(byte[] h1, byte[] h2)
            {
                if (h1.Length != HASH_LENGTH || h2.Length != HASH_LENGTH)
                {
                    throw new ArgumentException("Invalid hash");
                }

                for (var i = h1.Length - 1; i >= 0; i--)
                {
                    var byteEqual = (sbyte)h1[i] - (sbyte)h2[i];
                    if (byteEqual != 0)
                    {
                        return byteEqual;
                    }
                }

                return 0;
            }
        }

        static void Main()
        {
            // Get a digest
            GetDigestRequest getDigestRequest = new GetDigestRequest
            {
                Name = ledgerName
            };
            GetDigestResponse getDigestResponse =
client.GetDigestAsync(getDigestRequest).Result;

            // expectedDigest is the buffer we will use later to compare against our
calculated digest
            MemoryStream digest = getDigestResponse.Digest;
            byte[] expectedDigest = digest.ToArray();

            // Retrieve info for the given vin's document revisions
            var result = driver.Execute(txn => {
                string query = $"SELECT blockAddress, hash, metadata.id FROM
_ql_committed_{tableName} WHERE data.VIN = '{vin}'";
                return txn.Execute(query);
            });
```

```
            Console.WriteLine($"Verifying document revisions for vin '{vin}' in
table '{tableName}' in ledger '{ledgerName}'");

            foreach (IIonValue ionValue in result)
            {
                IIonStruct ionStruct = ionValue;

                // Get the requested fields
                IIonValue blockAddress = ionStruct.GetField("blockAddress");
                IIonBlob hash = ionStruct.GetField("hash");
                String metadataId = ionStruct.GetField("id").StringValue;

                Console.WriteLine($"Verifying document revision for id
'{metadataId}'");

                // Use an Ion Reader to convert block address to text
                IIonReader reader = IonReaderBuilder.Build(blockAddress);
                StringWriter sw = new StringWriter();
                IIonWriter textWriter = IonTextWriterBuilder.Build(sw);
                textWriter.WriteValues(reader);
                string blockAddressText = sw.ToString();

                // Submit a request for the revision
                GetRevisionRequest revisionRequest = new GetRevisionRequest
                {
                    Name = ledgerName,
                    BlockAddress = new ValueHolder
                    {
                        IonText = blockAddressText
                    },
                    DocumentId = metadataId,
                    DigestTipAddress = getDigestResponse.DigestTipAddress
                };

                // Get a response back
                GetRevisionResponse revisionResponse =
client.GetRevisionAsync(revisionRequest).Result;

                string proofText = revisionResponse.Proof.IonText;
                IIonDatagram proofValue = IonLoader.Default.Load(proofText);

                byte[] documentHash = hash.Bytes().ToArray();
                foreach (IIonValue proofHash in proofValue.GetElementAt(0))
                {
```

```
                    // Calculate the digest
                    documentHash = Dot(documentHash, proofHash.Bytes().ToArray());
                }

                bool verified = expectedDigest.SequenceEqual(documentHash);

                if (verified)
                {
                    Console.WriteLine($"Successfully verified document revision for
id '{metadataId}'!");
                }
                else
                {
                    Console.WriteLine($"Document revision for id '{metadataId}'
verification failed!");
                    return;
                }

                // Submit a request for the block
                GetBlockRequest getBlockRequest = new GetBlockRequest
                {
                    Name = ledgerName,
                    BlockAddress = new ValueHolder
                    {
                        IonText = blockAddressText
                    },
                    DigestTipAddress = getDigestResponse.DigestTipAddress
                };

                // Get a response back
                GetBlockResponse getBlockResponse =
client.GetBlockAsync(getBlockRequest).Result;

                string blockText = getBlockResponse.Block.IonText;
                IIonDatagram blockValue = IonLoader.Default.Load(blockText);

                // blockValue is a IonDatagram, and the first value is an IonStruct
containing the blockHash
                byte[] blockHash =
blockValue.GetElementAt(0).GetField("blockHash").Bytes().ToArray();

                proofText = getBlockResponse.Proof.IonText;
                proofValue = IonLoader.Default.Load(proofText);
```

```
                foreach (IIonValue proofHash in proofValue.GetElementAt(0))
                {
                    // Calculate the digest
                    blockHash = Dot(blockHash, proofHash.Bytes().ToArray());
                }

                verified = expectedDigest.SequenceEqual(blockHash);

                if (verified)
                {
                    Console.WriteLine($"Block address '{blockAddressText}'
 successfully verified!");
                }
                else
                {
                    Console.WriteLine($"Block address '{blockAddressText}'
 verification failed!");
                }
            }
        }
    }
}
```

Go

```
package main

import (
    "context"
    "crypto/sha256"
    "errors"
    "fmt"
    "reflect"

    "github.com/amzn/ion-go/ion"
    "github.com/aws/aws-sdk-go/aws"
    AWSSession "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/qldb"
    "github.com/aws/aws-sdk-go/service/qldbsession"
    "github.com/awslabs/amazon-qldb-driver-go/qldbdriver"
)

const (
```

```
    hashLength = 32
    ledgerName = "vehicle-registration"
    tableName  = "VehicleRegistration"
    vin        = "KM8SRDHF6EU074761"
)

// Takes two hashes, sorts them, concatenates them, and then returns the hash of the
 concatenated array.
func dot(h1, h2 []byte) ([]byte, error) {
    compare, err := hashComparator(h1, h2)
    if err != nil {
        return nil, err
    }

    var concatenated []byte
    if compare < 0 {
        concatenated = append(h1, h2...)
    } else {
        concatenated = append(h2, h1...)
    }

    newHash := sha256.Sum256(concatenated)
    return newHash[:], nil
}

func hashComparator(h1 []byte, h2 []byte) (int16, error) {
    if len(h1) != hashLength || len(h2) != hashLength {
        return 0, errors.New("invalid hash")
    }
    for i := range h1 {
        // Reverse index for little endianness
        index := hashLength - 1 - i

        // Handle byte being unsigned and overflow
        h1Int := int16(h1[index])
        h2Int := int16(h2[index])
        if h1Int > 127 {
            h1Int = 0 - (256 - h1Int)
        }
        if h2Int > 127 {
            h2Int = 0 - (256 - h2Int)
        }

        difference := h1Int - h2Int
```

```
        if difference != 0 {
            return difference, nil
        }
    }
    return 0, nil
}

func main() {
    driverSession := AWSSession.Must(AWSSession.NewSession(aws.NewConfig()))
    qldbSession := qldbsession.New(driverSession)
    driver, err := qldbdriver.New(ledgerName, qldbSession, func(options
 *qldbdriver.DriverOptions) {})
    if err != nil {
        panic(err)
    }
    client := qldb.New(driverSession)

    // Get a digest
    currentLedgerName := ledgerName
    input := qldb.GetDigestInput{Name: &currentLedgerName}
    digestOutput, err := client.GetDigest(&input)
    if err != nil {
        panic(err)
    }

    // expectedDigest is the buffer we will later use to compare against our
 calculated digest
    expectedDigest := digestOutput.Digest

    // Retrieve info for the given vin's document revisions
    result, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
        statement := fmt.Sprintf(
                "SELECT blockAddress, hash, metadata.id FROM _ql_committed_%s WHERE
 data.VIN = '%s'",
                tableName,
                vin)
        result, err := txn.Execute(statement)
        if err != nil {
            return nil, err
        }

        results := make([]map[string]interface{}, 0)
```

```
        // Convert the result set into a map
        for result.Next(txn) {
            var doc map[string]interface{}
            err := ion.Unmarshal(result.GetCurrentData(), &doc)
            if err != nil {
                return nil, err
            }
            results = append(results, doc)
        }
        return results, nil
    })
    if err != nil {
        panic(err)
    }
    resultSlice := result.([]map[string]interface{})

    fmt.Printf("Verifying document revisions for vin '%s' in table '%s' in ledger
'%s'\n", vin, tableName, ledgerName)

    for _, value := range resultSlice {
        // Get the requested fields
        ionBlockAddress, err := ion.MarshalText(value["blockAddress"])
        if err != nil {
            panic(err)
        }
        blockAddress := string(ionBlockAddress)
        metadataId := value["id"].(string)
        documentHash := value["hash"].([]byte)

        fmt.Printf("Verifying document revision for id '%s'\n", metadataId)

        // Submit a request for the revision
        revisionInput := qldb.GetRevisionInput{
            BlockAddress:     &qldb.ValueHolder{IonText: &blockAddress},
            DigestTipAddress: digestOutput.DigestTipAddress,
            DocumentId:       &metadataId,
            Name:             &currentLedgerName,
        }

        // Get a result back
        revisionOutput, err := client.GetRevision(&revisionInput)
        if err != nil {
            panic(err)
        }
```

```
        proofText := revisionOutput.Proof.IonText

        // Use ion.Reader to iterate over the proof's node hashes
        reader := ion.NewReaderString(*proofText)
        // Enter the struct containing node hashes
        reader.Next()
        if err := reader.StepIn(); err != nil {
            panic(err)
        }

        // Going through nodes and calculate digest
        for reader.Next() {
            val, _ := reader.ByteValue()
            documentHash, err = dot(documentHash, val)
        }

        // Compare documentHash with the expected digest
        verified := reflect.DeepEqual(documentHash, expectedDigest)

        if verified {
            fmt.Printf("Successfully verified document revision for id '%s'!\n",
metadataId)
        } else {
            fmt.Printf("Document revision for id '%s' verification failed!\n",
metadataId)
            return
        }

        // Submit a request for the block
        blockInput := qldb.GetBlockInput{
            Name:             &currentLedgerName,
            BlockAddress:     &qldb.ValueHolder{IonText: &blockAddress},
            DigestTipAddress: digestOutput.DigestTipAddress,
        }

        // Get a result back
        blockOutput, err := client.GetBlock(&blockInput)
        if err != nil {
            panic(err)
        }

        proofText = blockOutput.Proof.IonText
```

```
        block := new(map[string]interface{})
        err = ion.UnmarshalString(*blockOutput.Block.IonText, block)
        if err != nil {
            panic(err)
        }

        blockHash := (*block)["blockHash"].([]byte)

        // Use ion.Reader to iterate over the proof's node hashes
        reader = ion.NewReaderString(*proofText)
        // Enter the struct containing node hashes
        reader.Next()
        if err := reader.StepIn(); err != nil {
            panic(err)
        }

        // Going through nodes and calculate digest
        for reader.Next() {
            val, err := reader.ByteValue()
            if err != nil {
                panic(err)
            }
            blockHash, err = dot(blockHash, val)
        }

        // Compare blockHash with the expected digest
        verified = reflect.DeepEqual(blockHash, expectedDigest)

        if verified {
            fmt.Printf("Block address '%s' successfully verified!\n", blockAddress)
        } else {
            fmt.Printf("Block address '%s' verification failed!\n", blockAddress)
            return
        }
    }
}
```

Node.js

```
import { QldbDriver, Result, TransactionExecutor} from "amazon-qldb-driver-nodejs";
import { QLDB } from "aws-sdk"
import { GetBlockRequest, GetBlockResponse, GetDigestRequest, GetDigestResponse,
 GetRevisionRequest, GetRevisionResponse } from "aws-sdk/clients/qldb";
```

```
import { createHash } from "crypto";
import { dom, dumpText, load } from "ion-js"

const ledgerName: string = "vehicle-registration";
const tableName: string = "VehicleRegistration";
const vin: string = "KM8SRDHF6EU074761";
const driver: QldbDriver = new QldbDriver(ledgerName);
const qldbClient: QLDB = new QLDB();
const HASH_SIZE = 32;

/**
 * Takes two hashes, sorts them, concatenates them, and calculates a digest based on
 the concatenated hash.
 * @param h1 Byte array containing one of the hashes to compare.
 * @param h2 Byte array containing one of the hashes to compare.
 * @returns The digest calculated from the concatenated hash values.
 */
function dot(h1: Uint8Array, h2: Uint8Array): Uint8Array {
    if (h1.length === 0) {
        return h2;
    }
    if (h2.length === 0) {
        return h1;
    }

    const newHashLib = createHash("sha256");

    let concatenated: Uint8Array;
    if (hashComparator(h1, h2) < 0) {
        concatenated = concatenate(h1, h2);
    } else {
        concatenated = concatenate(h2, h1);
    }
    newHashLib.update(concatenated);
    return newHashLib.digest();
}

/**
 * Compares two hashes by their **signed** byte values in little-endian order.
 * @param hash1 The hash value to compare.
 * @param hash2 The hash value to compare.
 * @returns Zero if the hash values are equal, otherwise return the difference of
 the first pair of non-matching
 *          bytes.
```

```
 * @throws RangeError When the hash is not the correct hash size.
 */
function hashComparator(hash1: Uint8Array, hash2: Uint8Array): number {
    if (hash1.length !== HASH_SIZE || hash2.length !== HASH_SIZE) {
        throw new RangeError("Invalid hash.");
    }
    for (let i = hash1.length-1; i >= 0; i--) {
        const difference: number = (hash1[i]<<24 >>24) - (hash2[i]<<24 >>24);
        if (difference !== 0) {
            return difference;
        }
    }
    return 0;
}

/**
 * Helper method that concatenates two Uint8Array.
 * @param arrays List of arrays to concatenate, in the order provided.
 * @returns The concatenated array.
 */
function concatenate(...arrays: Uint8Array[]): Uint8Array {
    let totalLength = 0;
    for (const arr of arrays) {
        totalLength += arr.length;
    }
    const result = new Uint8Array(totalLength);
    let offset = 0;
    for (const arr of arrays) {
        result.set(arr, offset);
        offset += arr.length;
    }
    return result;
}

/**
 * Helper method that checks for equality between two Uint8Array.
 * @param expected Byte array containing one of the hashes to compare.
 * @param actual Byte array containing one of the hashes to compare.
 * @returns Boolean indicating equality between the two Uint8Array.
 */
function isEqual(expected: Uint8Array, actual: Uint8Array): boolean {
    if (expected === actual) return true;
    if (expected == null || actual == null) return false;
    if (expected.length !== actual.length) return false;
```

```
    for (let i = 0; i < expected.length; i++) {
        if (expected[i] !== actual[i]) {
            return false;
        }
    }
    return true;
}

const main = async function (): Promise<void> {
    // Get a digest
    const getDigestRequest: GetDigestRequest = {
        Name: ledgerName
    };
    const getDigestResponse: GetDigestResponse = await
 qldbClient.getDigest(getDigestRequest).promise();

    // expectedDigest is the buffer we will later use to compare against our
 calculated digest
    const expectedDigest: Uint8Array = <Uint8Array>getDigestResponse.Digest;

    const result: dom.Value[] = await driver.executeLambda(async (txn:
 TransactionExecutor): Promise<dom.Value[]> => {
        const query: string = `SELECT blockAddress, hash, metadata.id FROM
 _ql_committed_${tableName} WHERE data.VIN = '${vin}'`;
        const queryResult: Result = await txn.execute(query);
        return queryResult.getResultList();
    });

    console.log(`Verifying document revisions for vin '${vin}' in table
 '${tableName}' in ledger '${ledgerName}'`);

    for (let value of result) {
        // Get the requested fields
        const blockAddress: dom.Value = value.get("blockAddress");
        const hash: dom.Value = value.get("hash");
        const metadataId: string = value.get("id").stringValue();

        console.log(`Verifying document revision for id '${metadataId}'`);

        // Submit a request for the revision
        const revisionRequest: GetRevisionRequest = {
            Name: ledgerName,
            BlockAddress: {
```

```
                IonText: dumpText(blockAddress)
            },
            DocumentId: metadataId,
            DigestTipAddress: getDigestResponse.DigestTipAddress
        };

        // Get a response back
        const revisionResponse: GetRevisionResponse = await
qldbClient.getRevision(revisionRequest).promise();

        let proofValue: dom.Value = load(revisionResponse.Proof.IonText);

        let documentHash: Uint8Array = hash.uInt8ArrayValue();
        proofValue.elements().forEach((proofHash: dom.Value) => {
            // Calculate the digest
            documentHash = dot(documentHash, proofHash.uInt8ArrayValue());
        });

        let verified: boolean = isEqual(expectedDigest, documentHash);

        if (verified) {
            console.log(`Successfully verified document revision for id
'${metadataId}'!`);
        } else {
            console.log(`Document revision for id '${metadataId}' verification
failed!`);
            return;
        }

        // Submit a request for the block
        const getBlockRequest: GetBlockRequest = {
            Name: ledgerName,
            BlockAddress: {
                IonText: dumpText(blockAddress)
            },
            DigestTipAddress: getDigestResponse.DigestTipAddress
        };

        // Get a response back
        const getBlockResponse: GetBlockResponse = await
qldbClient.getBlock(getBlockRequest).promise();

        const blockValue: dom.Value = load(getBlockResponse.Block.IonText)
        let blockHash: Uint8Array = blockValue.get("blockHash").uInt8ArrayValue();
```

```
        proofValue = load(getBlockResponse.Proof.IonText);

        proofValue.elements().forEach((proofHash: dom.Value) => {
            // Calculate the digest
            blockHash = dot(blockHash, proofHash.uInt8ArrayValue());
        });

        verified = isEqual(expectedDigest, blockHash);

        if (verified) {
            console.log(`Block address '${dumpText(blockAddress)}' successfully
 verified!`);
        } else {
            console.log(`Block address '${dumpText(blockAddress)}' verification
 failed!`);
        }
    }
};

if (require.main === module) {
    main();
}
```

Python

```
from amazon.ion.simpleion import dumps, loads
from array import array
from boto3 import client
from functools import reduce
from hashlib import sha256
from pyqldb.driver.qldb_driver import QldbDriver

ledger_name = 'vehicle-registration'
table_name = 'VehicleRegistration'
vin = 'KM8SRDHF6EU074761'
qldb_client = client('qldb')
hash_length = 32


def query_doc_revision(txn):
    query = "SELECT blockAddress, hash, metadata.id FROM _ql_committed_{} WHERE
 data.VIN = '{}'".format(table_name, vin)
```

```
    return txn.execute_statement(query)


def block_address_to_dictionary(ion_dict):
    """
    Convert a block address from IonPyDict into a dictionary.
    Shape of the dictionary must be: {'IonText': "{strandId: <"strandId">,
 sequenceNo: <sequenceNo>}"}

    :type ion_dict: :py:class:`amazon.ion.simple_types.IonPyDict`/str
    :param ion_dict: The block address value to convert.

    :rtype: dict
    :return: The converted dict.
    """
    block_address = {'IonText': {}}
    if not isinstance(ion_dict, str):
        py_dict = '{{strandId: "{}", sequenceNo:{}}}'.format(ion_dict['strandId'],
 ion_dict['sequenceNo'])
        ion_dict = py_dict
    block_address['IonText'] = ion_dict
    return block_address


def dot(hash1, hash2):
    """
    Takes two hashes, sorts them, concatenates them, and then returns the
    hash of the concatenated array.

    :type hash1: bytes
    :param hash1: The hash value to compare.

    :type hash2: bytes
    :param hash2: The hash value to compare.

    :rtype: bytes
    :return: The new hash value generated from concatenated hash values.
    """
    if len(hash1) != hash_length or len(hash2) != hash_length:
        raise ValueError('Illegal hash.')

    hash_array1 = array('b', hash1)
    hash_array2 = array('b', hash2)
```

```
    difference = 0
    for i in range(len(hash_array1) - 1, -1, -1):
        difference = hash_array1[i] - hash_array2[i]
        if difference != 0:
            break

    if difference < 0:
        concatenated = hash1 + hash2
    else:
        concatenated = hash2 + hash1

    new_hash_lib = sha256()
    new_hash_lib.update(concatenated)
    new_digest = new_hash_lib.digest()
    return new_digest


# Get a digest
get_digest_response = qldb_client.get_digest(Name=ledger_name)

# expected_digest is the buffer we will later use to compare against our calculated
 digest
expected_digest = get_digest_response.get('Digest')
digest_tip_address = get_digest_response.get('DigestTipAddress')

qldb_driver = QldbDriver(ledger_name=ledger_name)

# Retrieve info for the given vin's document revisions
result = qldb_driver.execute_lambda(query_doc_revision)

print("Verifying document revisions for vin '{}' in table '{}' in ledger
 '{}'".format(vin, table_name, ledger_name))

for value in result:
    # Get the requested fields
    block_address = value['blockAddress']
    document_hash = value['hash']
    metadata_id = value['id']

    print("Verifying document revision for id '{}'".format(metadata_id))

    # Submit a request for the revision and get a result back
    proof_response = qldb_client.get_revision(Name=ledger_name,
```

```
BlockAddress=block_address_to_dictionary(block_address),
                                                DocumentId=metadata_id,
                                                DigestTipAddress=digest_tip_address)

   proof_text = proof_response.get('Proof').get('IonText')
   proof_hashes = loads(proof_text)

   # Calculate digest
   calculated_digest = reduce(dot, proof_hashes, document_hash)

   verified = calculated_digest == expected_digest
   if verified:
       print("Successfully verified document revision for id
'{}'!".format(metadata_id))
   else:
       print("Document revision for id '{}' verification
failed!".format(metadata_id))

   # Submit a request for the block and get a result back
   block_response = qldb_client.get_block(Name=ledger_name,
BlockAddress=block_address_to_dictionary(block_address),
                                          DigestTipAddress=digest_tip_address)

   block_text = block_response.get('Block').get('IonText')
   block = loads(block_text)

   block_hash = block.get('blockHash')

   proof_text = block_response.get('Proof').get('IonText')
   proof_hashes = loads(proof_text)

   # Calculate digest
   calculated_digest = reduce(dot, proof_hashes, block_hash)

   verified = calculated_digest == expected_digest
   if verified:
       print("Block address '{}' successfully
verified!".format(dumps(block_address,
                                                                    binary=False,

omit_version_marker=True)))
   else:
```

```
        print("Block address '{}' verification failed!".format(block_address))
```

Häufige Fehler bei der Überprüfung

In diesem Abschnitt werden Laufzeitfehler beschrieben, die von Amazon QLDB für Verifizierungsanforderungen ausgelöst werden.

Im Folgenden finden Sie eine Liste mit allgemeinen Ausnahmen, die vom Service zurückgegeben werden. Jede Ausnahme enthält die spezifische Fehlermeldung, gefolgt von den API-Operationen, die sie auslösen können, eine kurze Beschreibung und Vorschläge für mögliche Lösungen.

IllegalArgumentException

Nachricht: Der angegebene Ion-Wert ist ungültig und kann nicht analysiert werden.

API-Operationen: `GetDigest, GetBlock, GetRevision`

Stellen Sie sicher, dass Sie einen gültigen Amazon Ion-Wert bereitstellen, bevor Sie Ihre Anfrage erneut stellen.

IllegalArgumentException

Meldung: Die angegebene Blockadresse ist ungültig.

API-Operationen: `GetDigest, GetBlock, GetRevision`

Stellen Sie sicher, dass Sie eine gültige Block-Adresse bereitstellen, bevor Sie Ihre Anfrage erneut stellen. Ein Block-Adresse ist eine Amazon Ion-Struktur, die über zwei Felder verfügt: `strandId` und `sequenceNo`.

Beispiel: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

IllegalArgumentException

Meldung: Die Sequenznummer der angegebenen Digest-Tipp-Adresse geht über den letzten festgeschriebenen Datensatz des hinaus.

API-Operationen: `GetDigest, GetBlock, GetRevision`

Die Digest-Tip-Adresse, die Sie angeben, muss über eine Sequenznummer verfügen, die kleiner oder gleich der Sequenznummer des zuletzt übertragenen Datensatzes der Journal-Strähne ist.

Bevor Sie Ihre Anforderung wiederholen, stellen Sie sicher, dass Sie eine Digest-Tip-Adresse mit einer gültigen Sequenznummer bereitstellen.

IllegalArgumentException

Meldung: The Strand ID of the provided block address is not valid.

API-Operationen: `GetDigest, GetBlock, GetRevision`

Die von Ihnen angegebene Block-Adresse muss eine Strähnen-ID aufweisen, die der Strähnen-ID des Journals entspricht. Bevor Sie Ihre Anforderung wiederholen, stellen Sie sicher, dass Sie eine Block-Adresse mit einer gültigen Strähnen-ID bereitstellen.

IllegalArgumentException

Meldung: Die Sequenznummer der angegebenen Blockadresse geht über den letzten festgeschriebenen Datensatz des hinaus.

API-Operationen: `GetBlock, GetRevision`

Die Block-Adresse, die Sie angeben, muss über eine Sequenznummer verfügen, die kleiner oder gleich der Sequenznummer des zuletzt übertragenen Datensatzes der Strähne ist. Bevor Sie Ihre Anforderung wiederholen, stellen Sie sicher, dass Sie eine Block-Adresse mit einer gültigen Sequenznummer.

IllegalArgumentException

Nachricht: Die Strähnen-ID der bereitgestellten Block-Adresse muss der Strähnen-ID der bereitgestellten Digest-Tip-Adresse entsprechen.

API-Operationen: `GetBlock, GetRevision`

Sie können ein Dokument nur überprüfen oder blockieren, wenn es in derselben Journal-Strähne vorhanden ist wie das von Ihnen angegebene Digest-Journal.

IllegalArgumentException

Nachricht: Die Sequenznummer der bereitgestellten Block-Adresse darf nicht größer sein als die Sequenznummer der bereitgestellten Digest-Tip-Adresse.

API-Operationen: `GetBlock, GetRevision`

Sie können eine Dokumentrevision nur überprüfen oder blockieren, wenn sie von dem von Ihnen bereitgestellten Digest abgedeckt wird. Dies bedeutet, dass sie vor der Digest-Tip-Adresse an das Journal übertragen wurde.

IllegalArgumentException

Nachricht: Die angegebene Dokument-ID wurde im Block an der angegebenen Block-Adresse nicht gefunden.

API-Operation: `GetRevision`

Die Dokument-ID, die Sie angeben, muss in der von Ihnen angegebenen Block-Adresse vorhanden sein. Bevor Sie Ihre Anforderung wiederholen, stellen Sie sicher, dass diese beiden Parameter konsistent sind.

Exportieren von Journaldaten aus Amazon QLDB

Amazon QLDB verwendet ein unveränderliches Transaktionsprotokoll, das als Journal bezeichnet wird, für die Datenspeicherung. Das Journal verfolgt jede Änderung an Ihren zugesagten Daten und führt einen vollständigen und überprüfbaren Verlauf der Änderungen im Laufe der Zeit.

Sie können zu verschiedenen Zwecken auf den Inhalt des Journals im Ledger zugreifen, darunter Analyse, Prüfung, Datenaufbewahrung, Verifizierung und Export in andere Systeme. In den folgenden Themen wird beschrieben, wie Sie Journalblöcke aus Ihrem Ledger in einen Amazon Simple Storage Service (Amazon S3)-Bucket in Ihrem exportieren AWS-Konto. Ein Journalexportauftrag schreibt Ihre Daten in Amazon S3 als Objekte im Text- oder Binärdarstellung des Amazon-Ion-Formats oder im JSON-Lines-Textformat.

Im Format JSON Lines ist jeder Block in einem exportierten Datenobjekt ein gültiges JSON-Objekt, das durch eine Zeilenumbruch getrennt wird. Sie können dieses Format verwenden, um JSON-Exporte direkt in Analysetools wie Amazon Athena und zu integrieren, AWS Glue da diese Services JSON mit Zeilenumbruch automatisch analysieren können. Weitere Informationen zum Format finden Sie unter JSON-Zeilen.

Weitere Informationen zu Amazon S3 finden Sie im Benutzerhandbuch für Amazon Simple Storage Service.

> **Note**
>
> Wenn Sie JSON als Ausgabeformat Ihres Exportauftrags angeben, konvertiert QLDB die Ion-Journaldaten in Ihren exportierten Datenobjekten nach unten in JSON. Weitere Informationen finden Sie unter Abwärtskonvertieren zu JSON.

Themen

Anfordern eines Journalexports in QLDB

Amazon QLDB stellt eine API bereit, um einen Export Ihrer Journalblöcke für einen bestimmten Zeitraum und ein bestimmtes Amazon S3-Bucket-Ziel anzufordern. Ein Journalexportauftrag kann die Datenobjekte entweder im Text- oder in der binären Darstellung des Amazon-Ion-Formats oder im JSON-Lines-Textformat schreiben. Sie können die AWS Management Console, ein AWS SDK oder die AWS Command Line Interface (AWS CLI) verwenden, um einen Exportauftrag zu erstellen.

Themen

AWS Management Console

Gehen Sie wie folgt vor, um eine Journalexportanforderung in QLDB über die QLDB-Konsole zu senden.

So beantragen Sie einen Export (Konsole)

1. Melden Sie sich bei der an AWS Management Consoleund öffnen Sie die Amazon-QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich die Option Export (Exportieren) aus.
3. Wählen Sie Create export job (Exportauftrag erstellen) aus.
4. Geben Sie auf der Seite Create export job (Exportauftrag erstellen) die folgenden Exporteinstellungen ein:

   - Ledger – Das Ledger, dessen Journalblöcke Sie exportieren möchten.
   - Startdatum und -zeit – Der inklusive Startzeitstempel in UTC (Coordinated Universal Time) des Bereichs der zu exportierenden Journalblöcke. Dieser Zeitstempel muss früher als das End date and time (Enddatum und -uhrzeit) sein. Wenn Sie einen Startzeitstempel angeben, der älter als der des Ledgers ist`CreationDateTime`, verwendet QLDB standardmäßig den des Ledgers`CreationDateTime`.
   - Enddatum und -zeit – Der exklusive Endzeitstempel (UTC) des Bereichs der zu exportierenden Journalblöcke. Dieses Datum und diese Uhrzeit dürfen nicht in der Zukunft liegen.

- Ziel für Journalblöcke – Der Amazon S3-Bucket und der Präfixname, in den Ihr Exportauftrag die Datenobjekte schreibt. Verwenden Sie das folgende Amazon S3-URI-Format.

  ```
  s3://DOC-EXAMPLE-BUCKET/prefix/
  ```

  Geben Sie einen S3-Bucket-Namen und einen optionalen Präfixnamen für die Ausgabeobjekte an. Im Folgenden wird ein Beispiel gezeigt.

  ```
  s3://DOC-EXAMPLE-BUCKET/journalExport/
  ```

  Der Bucket-Name und das Präfix müssen beide den Amazon S3-Benennungsregeln und -konventionen entsprechen. Weitere Informationen zur Bucket-Benennung finden Sie unter Bucket-Einschränkungen und -Einschränkungen im Amazon S3-Entwicklerhandbuch. Weitere Informationen zu Schlüsselnamenpräfixen finden Sie unter Objektschlüssel und Metadaten.

  > **Note**
  >
  > Regionsübergreifende Exporte werden nicht unterstützt. Der angegebene Amazon S3-Bucket muss sich in derselben AWS-Region wie Ihr Ledger befinden.

- S3-Verschlüsselung – Die Verschlüsselungseinstellungen, die von Ihrem Exportauftrag verwendet werden, um Daten in einen Amazon S3-Bucket zu schreiben. Weitere Informationen zu serverseitigen Verschlüsselungsoptionen in Amazon S3 finden Sie unter Schützen von Daten mithilfe serverseitiger Verschlüsselung im Amazon-S3-EntwicklerhandbuchAmazon S3.
  - Bucket-Standardverschlüsselung – Verwenden Sie die Standardverschlüsselungseinstellungen des angegebenen Amazon S3-Buckets.
  - AES-256 – Verwenden Sie die serverseitige Verschlüsselung mit von Amazon S3 verwalteten Schlüsseln (SSE-S3).
  - AWS-KMS – Verwenden Sie die serverseitige Verschlüsselung mit von AWS KMS verwalteten Schlüsseln (SSE-KMS).

    Wenn Sie diesen Typ zusammen mit der Option Anderen auswählen AWS KMS key auswählen wählen, müssen Sie auch einen KMS-Schlüssel mit symmetrischer Verschlüsselung im folgenden Amazon-Ressourcennamen (ARN)-Format angeben.

    ```
    arn:aws:kms:aws-region:account-id:key/key-id
    ```

- Servicezugriff – Die IAM-Rolle, die QLDB-Schreibberechtigungen in Ihrem Amazon S3-Bucket gewährt. Falls zutreffend, muss die IAM-Rolle auch QLDB-Berechtigungen zur Verwendung Ihres KMS-Schlüssels erteilen.

  Um beim Anfordern eines Journalexports eine Rolle an QLDB zu übergeben, müssen Sie über Berechtigungen zum Ausführen der *`iam:PassRole`* Aktion für die IAM-Rollenressource verfügen.

  - Erstellen und Verwenden einer neuen Servicerolle – Lassen Sie die Konsole eine neue Rolle für Sie mit den erforderlichen Berechtigungen für den angegebenen Amazon S3-Bucket erstellen.
  - Verwenden einer vorhandenen Servicerolle – Informationen zum manuellen Erstellen dieser Rolle in IAM finden Sie unter Exportberechtigungen.
- Ausgabeformat – Das Ausgabeformat Ihrer exportierten Journaldaten
  - Ion-Text – (Standard) Textdarstellung von Amazon Ion
  - Ion-Binärdatei – Binäre Darstellung von Amazon Ion
  - JSON – durch Zeilenumbrüche getrenntes JSON-Textformat

    Wenn Sie JSON wählen, konvertiert QLDB die Ion-Journaldaten in Ihren exportierten Datenobjekten nach unten in JSON. Weitere Informationen finden Sie unter Abwärtskonvertieren zu JSON.

5. Wenn Sie alle gewünschten Einstellungen vorgenommen haben, wählen Sie Create export job (Exportauftrag erstellen) aus.

   Die Zeit, die für die Durchführung des Exportauftrags benötigt wird, variiert je nach der Datengröße. Wenn die Übermittlung der Anforderung erfolgreich ist, kehrt die Konsole zur Hauptseite Export (Exportieren) zurück und listet Ihre Exportaufträge mit ihren aktuellen Status auf.

6. Sie können Ihre Exportobjekte in der Amazon S3-Konsole sehen.

   Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.

   Weitere Informationen zum Format dieser Ausgabeobjekte finden Sie unter Ausgabe des Journalexports in QLDB.

> **Note**
>
> Exportaufträge laufen sieben Tage nach deren Abschluss ab. Weitere Informationen finden Sie unter Ablauf des Exportauftrags.

QLDB-API

Sie können einen Journalexport auch über die Amazon-QLDB-API mit einem - AWS SDK oder anfordern AWS CLI. Die QLDB-API bietet die folgenden Operationen zur Verwendung durch Anwendungsprogramme:

- `ExportJournalToS3` – Exportiert Journalinhalte innerhalb eines Datums- und Zeitbereichs aus einem bestimmten Ledger in einen angegebenen Amazon S3-Bucket. Ein Exportauftrag kann die Daten entweder als Objekte im Text- oder Binärdarstellung des Amazon-Ion-Formats oder im JSON-Lines-Textformat schreiben.
- `DescribeJournalS3Export` – Gibt detaillierte Informationen zu einem Journalexportauftrag zurück. Die Ausgabe enthält den aktuellen Status, die Erstellungszeit und die Parameter Ihrer ursprünglichen Exportanforderung.
- `ListJournalS3Exports` – Gibt eine Liste der Auftragsbeschreibungen für den Journalexport für alle Ledger zurück, die dem aktuellen AWS-Konto und der Region zugeordnet sind. Die Ausgabe jeder Exportauftragsbeschreibung enthält dieselben Details, die von `DescribeJournalS3Export` zurückgegeben werden.
- `ListJournalS3ExportsForLedger` – Gibt eine Liste der Beschreibungen von Journalexportaufträgen für einen bestimmten Ledger zurück. Die Ausgabe jeder Exportauftragsbeschreibung enthält dieselben Details, die von `DescribeJournalS3Export` zurückgegeben werden.

Eine vollständige Beschreibung dieser API-Vorgänge finden Sie unter Amazon QLDB API-Referenz.

Informationen zum Exportieren von Journaldaten mit der finden Sie in der - AWS CLIBefehlsreferenz. AWS CLI

Beispielanwendung (Java)

Java-Codebeispiele für grundlegende Exportvorgänge finden Sie im GitHub Repository aws-samples/amazon-qldb-dmv-sample-java . Anweisungen zum Herunterladen und Installieren dieser

Beispielanwendung finden Sie unter Installation der Amazon QLDB Java-Beispielanwendung. Bevor Sie einen Export anfordern, stellen Sie sicher, dass Sie die Schritte 1–3 in der ausführen, Java-Anleitung um einen Beispiel-Ledger zu erstellen und ihn mit Beispieldaten zu laden.

Der Tutorial-Code in den folgenden Klassen enthält Beispiele für die Erstellung eines Exports, die Überprüfung des Status eines Exports und die Verarbeitung der Ausgabe eines Exports.

Klasse	Beschreibung
ExportJournal	Exportiert Journal-Blöcke aus dem `vehicle-registration` -Beispiel-Ledger für einen Zeitstempel-Bereich von vor 10 Minuten bis jetzt. Schreibt die Ausgabenobjekte in einen angegebenen S3-Bucket oder erstellt einen einzigartigen Bucket, wenn keiner angegeben ist.
DescribeJournalExport	Beschreibt einen Journalexportauftrag für eine angegebene `exportId` im `vehicle-registration` -Beispiel-Ledger.
ListJournalExports	Gibt eine Liste der Beschreibungen des Journal-Exportauftrags für den `vehicle-registration` -Beispiel-Ledger zurück.
ValidateQldbHashChain	Validiert die Hash-Kette des `vehicle-registration` -Beispiel-Ledgers mit einer gegebenen `exportId`. Wenn diese Option nicht angegeben ist, wird ein neuer Export für die Verwendung mit der Hash-Kettenvalidierung angefordert.

Ablauf des Exportauftrags

Abgeschlossene Journalexportaufträge unterliegen einem Aufbewahrungszeitraum von 7 Tagen. Sie werden nach Ablauf dieses Limits automatisch dauerhaft gelöscht. Dieser Ablaufzeitraum ist ein hartes Limit und kann nicht geändert werden.

Nachdem ein abgeschlossener Exportauftrag gelöscht wurde, können Sie die QLDB-Konsole oder die folgenden API-Operationen nicht mehr verwenden, um Metadaten über den Auftrag abzurufen:

- `DescribeJournalS3Export`
- `ListJournalS3Exports`
- `ListJournalS3ExportsForLedger`

Dieser Ablauf hat jedoch keine Auswirkungen auf die exportierten Daten. Alle Metadaten werden in den Manifestdateien aufbewahrt, die von Ihren Exporten geschrieben werden. Dieser Ablauf soll ein reibungsloseres Erlebnis für die API-Operationen bieten, die Journalexportaufträge auflisten. QLDB entfernt alte Exportaufträge, um sicherzustellen, dass Sie nur die letzten Exporte sehen, ohne mehrere Seiten von Aufträgen analysieren zu müssen.

Ausgabe des Journalexports in QLDB

Ein Amazon-QLDB-Journalexportauftrag schreibt zwei Manifestdateien zusätzlich zu den Datenobjekten, die Ihre Journalblöcke enthalten. Diese werden alle in dem Amazon S3-Bucket gespeichert, den Sie in Ihrer Exportanforderung angegeben haben. Die folgenden Abschnitte beschreiben das Format und den Inhalt der einzelnen Ausgabeobjekte.

> **Note**
>
> Wenn Sie JSON als Ausgabeformat Ihres Exportauftrags angeben, konvertiert QLDB die Amazon-Ion-Journaldaten in Ihren exportierten Datenobjekten nach unten in JSON. Weitere Informationen finden Sie unter Abwärtskonvertieren zu JSON.

Themen

- Manifestdateien
- Datenobjekte
- Abwärtskonvertieren zu JSON
- Exportieren der Prozessorbibliothek (Java)

Manifestdateien

Amazon QLDB erstellt zwei Manifestdateien im bereitgestellten S3-Bucket für jede Exportanforderung. Die ursprüngliche Manifestdatei wird erstellt, sobald Sie die Exportanforderung übermitteln. Die finale Manifestdatei wird geschrieben, nachdem der Exportvorgang abgeschlossen ist. Sie können diese Dateien verwenden, um den Status Ihrer Exportaufträge in Amazon S3 zu überprüfen.

Das Format für den Inhalt der Manifestdateien entspricht dem angeforderten Ausgabeformat für den Export.

Erstes Manifest

Die ursprüngliche Manifestdatei gibt an, dass Ihr Exportauftrag gestartet wurde. Sie enthält die Eingabeparameter, die Sie der Anforderung übergeben haben. Zusätzlich zum Amazon S3-Ziel und den Start- und Endzeitparametern für den Export enthält diese Datei auch einen `exportId`. ist `exportId` eine eindeutige ID, die QLDB jedem Exportauftrag zuweist.

Die Benennungskonvention für die Datei lautet wie folgt.

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.started.manifest
```

Im Folgenden finden Sie ein Beispiel für eine anfängliche Manifestdatei und ihren Inhalt im Ion-Textformat.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/8UyXulxccYLAsbN1aon7e4.started.manifest
```

```
{
  ledgerName:"my-example-ledger",
  exportId:"8UyXulxccYLAsbN1aon7e4",
  inclusiveStartTime:2019-04-15T00:00:00.000Z,
  exclusiveEndTime:2019-04-15T22:00:00.000Z,
  bucket:"DOC-EXAMPLE-BUCKET",
  prefix:"journalExport",
  objectEncryptionType:"NO_ENCRYPTION",
  outputFormat:"ION_TEXT"
}
```

Das anfängliche Manifest enthält `outputFormat` nur dann , wenn es in der Exportanforderung angegeben wurde. Wenn Sie das Ausgabeformat nicht angeben, verwenden die exportierten Daten standardmäßig das `ION_TEXT` Format .

Die DescribeJournalS3Export-API-Operation und der Inhaltstyp der exportierten Amazon S3-Objekte geben auch das Ausgabeformat an.

Endgültiges Manifest

Das finale Manifest gibt an, dass Ihr Exportauftrag für eine bestimmte Journal-Strähne abgeschlossen ist. Der Exportauftrag schreibt eine separate finale Manifestdatei für jeden Strähne.

> **Note**
>
> In Amazon QLDB ist ein eine Partition des Journals Ihres Ledgers. QLDB unterstützt derzeit nur Journale mit einem einzigen .

Die finale Manifestdatei enthält eine geordnete Liste der Daten- Objektschlüssel, die während des Exportvorgangs geschrieben wurden. Die Benennungskonvention für die Datei lautet wie folgt.

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.strandId.completed.manifest
```

ist `strandId` eine eindeutige ID, die QLDB dem zuweist. Im Folgenden finden Sie ein Beispiel für eine endgültige Manifestdatei und ihren Inhalt im Ion-Textformat.

```
s3://DOC-EXAMPLE-BUCKET/
journalExport/8UyXulxccYLAsbN1aon7e4.JdxjkR9bSYB5jMHWcI464T.completed.manifest
```

```
{
  keys:[
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.1-4.ion",
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.5-10.ion",
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.11-12.ion",
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.13-20.ion",
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.21-21.ion"
  ]
}
```

Datenobjekte

Amazon QLDB schreibt Journaldatenobjekte in den bereitgestellten Amazon S3-Bucket, entweder im Text- oder im Binärdarstellungsformat von Amazon Ion oder im JSON-Lines-Textformat.

Im Format JSON Lines ist jeder Block in einem exportierten Datenobjekt ein gültiges JSON-Objekt, das durch eine Zeilenumbruch getrennt wird. Sie können dieses Format verwenden, um JSON-Exporte direkt in Analysetools wie Amazon Athena und zu integrieren, AWS Glue da diese Services JSON mit Zeilenumbruch automatisch analysieren können. Weitere Informationen zum Format finden Sie unter JSON-Zeilen.

Datenobjektnamen

Ein Journalexportauftrag schreibt diese Datenobjekte mit der folgenden Benennungskonvention.

```
s3://DOC-EXAMPLE-BUCKET/prefix/yyyy/mm/dd/hh/strandId.startSn-endSn.ion|.json
```

- Die Ausgabedaten der einzelnen Exportaufträge sind in Datenblöcke aufgeteilt.
- `yyyy/mm/dd/hh` – Das Datum und die Uhrzeit, zu der Sie die Exportanforderung übermittelt haben. Objekte, die innerhalb derselben Stunde exportiert werden, sind unter demselben Amazon S3-Präfix gruppiert.
- `strandId` – Die eindeutige ID des bestimmten , der den zu exportierenden Journalblock enthält.
- `startSn-endSn` – Der Sequenznummernbereich, der im Objekt enthalten ist. Eine Sequenznummer gibt die Position eines Blocks innerhalb eines an.

Nehmen Sie zum Beispiel an, dass Sie den folgenden Pfad angeben.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/
```

Ihr Exportauftrag erstellt ein Amazon S3-Datenobjekt, das dem folgenden ähnelt. Dieses Beispiel zeigt einen Objektnamen im Ion-Format.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.1-5.ion
```

Inhalt des Datenobjekts

Jedes Datenobjekt enthält Journalblockobjekte im folgenden Format.

```
{
  blockAddress: {
    strandId: String,
    sequenceNo: Int
  },
  transactionId: String,
  blockTimestamp: Datetime,
  blockHash: SHA256,
  entriesHash: SHA256,
  previousBlockHash: SHA256,
  entriesHashList: [ SHA256 ],
  transactionInfo: {
    statements: [
      {
        //PartiQL statement object
      }
    ],
    documents: {
      //document-table-statement mapping object
    }
  },
  revisions: [
    {
      //document revision object
    }
  ]
}
```

Ein Block ist ein Objekt, das während einer Transaktion im Journal festgeschrieben wird. Ein Block enthält Transaktionsmetadaten zusammen mit Einträgen, die die Dokumentrevisionen darstellen, die in der Transaktion festgeschrieben wurden, sowie die PartiQL-Anweisungen, die diese festgeschrieben haben.

Im Folgenden finden Sie ein Beispiel für einen Block mit Beispieldaten im Ion-Textformat. Weitere Informationen zu den Feldern in einem Blockobjekt finden Sie unter Journalinhalte in Amazon QLDB.

> **Note**
>
> Dieses Blockbeispiel dient nur zu Informationszwecken. Die angezeigten Hashwerte sind keine tatsächlich berechneten Hashwerte.

```
{
  blockAddress:{
    strandId:"JdxjkR9bSYB5jMHWcI464T",
    sequenceNo:1234
  },
  transactionId:"D35qctdJRU1L1N2VhxbwSn",
  blockTimestamp:2019-10-25T17:20:21.009Z,
  blockHash:{{WYLOfZClk0lYWT3lUsSr0ONXh+Pw8MxxB+9zvTgSvlQ=}},
  entriesHash:{{xN9X96atkMvhvF3nEy6jMSVQzKjHJfz1H3bsNeg8GMA=}},
  previousBlockHash:{{IAfZ0h22ZjvcuHPSBCDy/6XNQTsqEmeY3GW0gBae8mg=}},
  entriesHashList:[
      {{F7rQIKCNn0vXVWPexilGfJn5+MCrtsSQqqVdlQxXpS4=}},
      {{C+L8gRhkzVcxt3qRJpw8w6hVEqA5A6ImGne+E7iHizo=}}
  ],
  transactionInfo:{
    statements:[
      {
        statement:"CREATE TABLE VehicleRegistration",
        startTime:2019-10-25T17:20:20.496Z,
        statementDigest:{{3jeSdejOgp6spJ8huZxDRUtp2fRXRqpOMtG43V0nXg8=}}
      },
      {
        statement:"CREATE INDEX ON VehicleRegistration (VIN)",
        startTime:2019-10-25T17:20:20.549Z,
        statementDigest:{{099D+5ZWDgA7r+aWeNUrWhc8ebBTXjgscq+mZ2dVibI=}}
      },
      {
        statement:"CREATE INDEX ON VehicleRegistration (LicensePlateNumber)",
        startTime:2019-10-25T17:20:20.560Z,
        statementDigest:{{B73tVJzVyVXicnH4n96NzU2L2JFY8e9Tjg895suWMew=}}
      },
      {
        statement:"INSERT INTO VehicleRegistration ?",
        startTime:2019-10-25T17:20:20.595Z,
        statementDigest:{{ggpon5qCXLo95K578YVhAD8ix0A0M5CcBx/W40Ey/Tk=}}
      }
    ],
    documents:{
      '8F0TPCmdNQ6JTRpiLj2TmW':{
        tableName:"VehicleRegistration",
        tableId:"BPxNiDQXCIB5l5F68KZoOz",
        statements:[3]
      }
```

```
    }
  },
  revisions:[
    {
      hash:{{FR1IWcWew0yw1TnRklo2YMF/qtwb7ohsu5FD8A4DSVg=}}
    },
    {
      blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:1234
      },
      hash:{{t8Hj6/VC4SBitxnvBqJbOmrGytF2XAA/1c0AoSq2NQY=}},
      data:{
        VIN:"1N4AL11D75C109151",
        LicensePlateNumber:"LEWISR261LL",
        State:"WA",
        City:"Seattle",
        PendingPenaltyTicketAmount:90.25,
        ValidFromDate:2017-08-21,
        ValidToDate:2020-05-11,
        Owners:{
          PrimaryOwner:{
            PersonId:"GddsXfIYfDlKCEprOLOwYt"
          },
          SecondaryOwners:[]
        }
      },
      metadata:{
        id:"8F0TPCmdNQ6JTRpiLj2TmW",
        version:0,
        txTime:2019-10-25T17:20:20.618Z,
        txId:"D35qctdJRU1L1N2VhxbwSn"
      }
    }
  ]
}
```

Im `revisions`-Feld enthalten einige Revisionsobjekte möglicherweise nur einen `hash`-Wert und keine anderen Attribute. Dabei handelt es sich um interne Systemrevisionen, die keine Benutzerdaten enthalten. Ein Exportauftrag enthält diese Revisionen in den jeweiligen Blöcken, da die Hashes dieser Revisionen Teil der vollständigen Hashkette des Journals sind. Die vollständige Hashkette ist für die kryptografische Verifizierung erforderlich.

Abwärtskonvertieren zu JSON

Wenn Sie JSON als Ausgabeformat Ihres Exportauftrags angeben, konvertiert QLDB die Amazon-Ion-Journaldaten in Ihren exportierten Datenobjekten in JSON. In bestimmten Fällen, in denen Ihre Daten die umfangreichen Ion-Typen verwenden, die in JSON nicht vorhanden sind, ist die Konvertierung von Ion in JSON jedoch verlustbehaftet.

Weitere Informationen zu Ion-zu-JSON-Konvertierungsregeln finden Sie unter Down-Converting to JSON im Amazon-Ion-Cookbook.

Exportieren der Prozessorbibliothek (Java)

QLDB bietet ein erweiterbares Framework für Java, das die Verarbeitung von Exporten in Amazon S3 optimiert. Diese Framework-Bibliothek erledigt das Lesen der Ausgabe eines Exports und das Iterieren der exportierten Blöcke in sequenzieller Reihenfolge. Informationen zur Verwendung dieses Exportprozessors finden Sie im GitHub Repository awslabs/amazon-qldb-export-processor-java .

Journalexportberechtigungen in QLDB

Bevor Sie eine Journalexportanforderung in Amazon QLDB einreichen, müssen Sie QLDB Schreibberechtigungen in Ihrem angegebenen Amazon S3-Bucket erteilen. Wenn Sie einen kundenverwalteten AWS KMS key als Objektverschlüsselungstyp für Ihren Amazon S3-Bucket auswählen, müssen Sie QLDB auch Berechtigungen zur Verwendung Ihres angegebenen symmetrischen Verschlüsselungsschlüssels erteilen. Amazon S3 unterstützt keine asymmetrischen KMS-Schlüssel.

Um Ihrem Exportauftrag die erforderlichen Berechtigungen bereitzustellen, können Sie QLDB eine IAM-Servicerolle mit den entsprechenden Berechtigungsrichtlinien übernehmen lassen. Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

> **Note**
>
> Um beim Anfordern eines Journalexports eine Rolle an QLDB zu übergeben, müssen Sie über Berechtigungen zum Ausführen der `iam:PassRole` Aktion für die IAM-Rollenressource verfügen. Dies gilt zusätzlich zur -`qldb:ExportJournalToS3`Berechtigung für die QLDB-Ledger-Ressource.

Informationen zum Steuern des Zugriffs auf QLDB mit IAM finden Sie unter Funktionsweise von Amazon QLDB mit IAM. Ein Beispiel für eine QLDB-Richtlinie finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

In diesem Beispiel erstellen Sie eine Rolle, die es QLDB ermöglicht, Objekte in Ihrem Namen in einen Amazon S3-Bucket zu schreiben. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

Wenn Sie zum AWS-Konto ersten Mal ein QLDB-Journal in Ihr exportieren, müssen Sie zunächst eine IAM-Rolle mit den entsprechenden Richtlinien erstellen, indem Sie wie folgt vorgehen. Oder Sie können die QLDB-Konsole verwenden, um die Rolle automatisch für Sie zu erstellen. Andernfalls können Sie eine Rolle auswählen, die Sie zuvor erstellt haben.

Themen

- Erstellen einer Berechtigungsrichtlinie
- Erstellen einer IAM-Rolle

Erstellen einer Berechtigungsrichtlinie

Führen Sie die folgenden Schritte aus, um eine Berechtigungsrichtlinie für einen QLDB-Journalexportauftrag zu erstellen. Dieses Beispiel zeigt eine Amazon S3-Bucket-Richtlinie, die QLDB Berechtigungen zum Schreiben von Objekten in den angegebenen Bucket gewährt. Falls zutreffend, zeigt das Beispiel auch eine Schlüsselrichtlinie, die es QLDB ermöglicht, Ihren KMS-Schlüssel mit symmetrischer Verschlüsselung zu verwenden.

Weitere Informationen zu Amazon S3-Bucket-Richtlinien finden Sie unter Verwenden von Bucket-Richtlinien und Benutzerrichtlinien im Benutzerhandbuch für Amazon Simple Storage Service. Weitere Informationen zu AWS KMS Schlüsselrichtlinien finden Sie unter Verwenden von Schlüsselrichtlinien in AWS KMS im AWS Key Management Service -Entwicklerhandbuch.

> **Note**
>
> Ihr Amazon S3-Bucket und Ihr KMS-Schlüssel müssen sich beide in derselben AWS-Region wie Ihr QLDB-Ledger befinden.

So verwenden Sie den JSON-Richtlinieneditor zum Erstellen einer Richtlinie

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste auf der linken Seite auf Policies (Richtlinien).

   Wenn Sie zum ersten Mal Policies (Richtlinien) auswählen, erscheint die Seite Welcome to Managed Policies (Willkommen bei verwalteten Richtlinien). Wählen Sie Get Started.
3. Wählen Sie oben auf der Seite Create policy (Richtlinie erstellen) aus.
4. Wählen Sie den Tab JSON.
5. Geben Sie ein JSON-Richtliniendokument ein.
   - Wenn Sie einen vom Kunden verwalteten KMS-Schlüssel für die Amazon S3-Objektverschlüsselung verwenden, verwenden Sie das folgende Beispielrichtliniendokument. Um diese Richtlinie zu verwenden, ersetzen Sie *DOC-EXAMPLE-BUCKET* , *us-east-1*, *123456789012* und *1234abcd-12ab-34cd-56ef-1234567890ab* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permission",
            "Action": [
                "s3:PutObjectAcl",
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        },
        {
            "Sid": "QLDBJournalExportKMSPermission",
            "Action": [ "kms:GenerateDataKey" ],
            "Effect": "Allow",
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

- Für andere Verschlüsselungstypen verwenden Sie das folgende Beispielrichtliniendokument. Um diese Richtlinie zu verwenden, ersetzen Sie *DOC-EXAMPLE-BUCKET* im Beispiel durch Ihren eigenen Amazon S3-Bucket-Namen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permission",
            "Action": [
                "s3:PutObjectAcl",
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

6. Wählen Sie Richtlinie prüfen.

> **Note**
>
> Sie können jederzeit zwischen den Registerkarten Visual editor (Visueller Editor) und JSON wechseln. Wenn Sie jedoch Änderungen vornehmen oder auf der Registerkarte Visueller Editor die Option Richtlinie überprüfen auswählen, strukturiert IAM möglicherweise Ihre Richtlinie neu, um sie für den visuellen Editor zu optimieren. Weitere Informationen finden Sie unter Richtlinienrestrukturierung im IAM-Benutzerhandbuch.

7. Geben Sie auf der Seite Review policy (Richtlinie überprüfen) unter Name einen Namen und unter Description (Beschreibung) eine optionale Beschreibung für die Richtlinie ein, die Sie erstellen. Überprüfen Sie unter Summary die Richtlinienzusammenfassung, um die Berechtigungen einzusehen, die von Ihrer Richtlinie gewährt werden. Wählen Sie dann Create policy aus, um Ihre Eingaben zu speichern.

Erstellen einer IAM-Rolle

Nachdem Sie eine Berechtigungsrichtlinie für Ihren QLDB-Journalexportauftrag erstellt haben, können Sie eine IAM-Rolle erstellen und ihr Ihre Richtlinie anfügen.

So erstellen Sie die Servicerolle für QLDB (IAM-Konsole)

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Klicken Sie im Navigationsbereich der IAM-Konsole auf Rollen, und wählen Sie dann Rolle erstellen.
3. Wählen Sie für Vertrauenswürdige Entität die Option AWS-Service aus.
4. Wählen Sie für Service oder Anwendungsfall QLDB und dann den QLDB-Anwendungsfall aus.
5. Wählen Sie Weiter aus.
6. Aktivieren Sie das Kontrollkästchen neben der Richtlinie, die Sie in den vorherigen Schritten erstellt haben.
7. (Optional) Legen Sie eine Berechtigungsgrenze fest. Dies ist ein erweitertes Feature, das für Servicerollen verfügbar ist, aber nicht für servicegebundene Rollen.

   a. Öffnen Sie den Abschnitt Berechtigungsgrenze festlegen und wählen Sie dann Berechtigungsgrenze verwenden aus, um die maximalen Rollenberechtigungen zu steuern.

      IAM enthält eine Liste der AWS von verwalteten und vom Kunden verwalteten Richtlinien in Ihrem Konto.

   b. Wählen Sie die Richtlinie aus, die für eine Berechtigungsgrenze verwendet werden soll.
8. Wählen Sie Weiter aus.
9. Geben Sie einen Rollennamen oder ein Suffix für den Rollennamen ein, um den Zweck der Rolle zu identifizieren.

> **Important**
>
> Beachten Sie beim Benennen einer Rolle Folgendes:
>
> - Rollennamen müssen innerhalb Ihres eindeutig sein AWS-Kontound können nicht durch Groß- und Kleinschreibung eindeutig gemacht werden.
>
>   Erstellen Sie beispielsweise keine Rollen mit den Namen **`PRODROLE`** und **`prodrole`**. Wenn ein Rollenname in einer Richtlinie oder als Teil eines ARN verwendet wird, wird die Groß- und Kleinschreibung beachtet. Wenn Kunden in der Konsole jedoch ein Rollenname angezeigt wird, z. B. während des Anmeldevorgangs, wird die Groß- und Kleinschreibung des Rollennamens nicht beachtet.

- Sie können den Namen der Rolle nach ihrer Erstellung nicht mehr bearbeiten, da andere Entitäten möglicherweise auf die Rolle verweisen.

10. (Optional) Geben Sie unter Beschreibung eine Beschreibung für die Rolle ein.
11. (Optional) Um die Anwendungsfälle und Berechtigungen für die Rolle zu bearbeiten, wählen Sie in den Abschnitten Schritt 1: Vertrauenswürdige Entitäten auswählen oder Schritt 2: Berechtigungen hinzufügen die Option Bearbeiten aus.
12. (Optional) Um die Rolle zu identifizieren, zu organisieren oder zu suchen, fügen Sie Tags als Schlüssel-Wert-Paare hinzu. Weitere Informationen zur Verwendung von Tags in IAM finden Sie unter Markieren von IAM-Ressourcen im IAM-Benutzerhandbuch.
13. Prüfen Sie die Rolle und klicken Sie dann auf Create Role (Rolle erstellen).

Das folgende JSON-Dokument ist ein Beispiel für eine Vertrauensrichtlinie, die es QLDB ermöglicht, eine IAM-Rolle mit bestimmten Berechtigungen anzunehmen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "qldb.amazonaws.com"
            },
            "Action": [ "sts:AssumeRole" ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:qldb:us-east-1:123456789012:*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

> **Note**
>
> Dieses Beispiel für eine Vertrauensrichtlinie zeigt, wie Sie die `aws:SourceAccount` globalen Bedingungskontextschlüssel `aws:SourceArn` und verwenden können, um das Confused-Deputy-Problem zu vermeiden. Mit dieser Vertrauensrichtlinie kann QLDB die Rolle `123456789012` nur für jede QLDB-Ressource im Konto übernehmen.
> Weitere Informationen finden Sie unter Serviceübergreifende Confused-Deputy-Prävention.

Nachdem Sie Ihre IAM-Rolle erstellt haben, kehren Sie zur QLDB-Konsole zurück und aktualisieren Sie die Seite Exportauftrag erstellen, damit sie Ihre neue Rolle finden kann.

Häufige Fehler beim Exportieren von Journalen

In diesem Abschnitt werden Laufzeitfehler beschrieben, die von Amazon QLDB für Journalexportanforderungen ausgelöst werden.

Im Folgenden finden Sie eine Liste mit allgemeinen Ausnahmen, die vom Service zurückgegeben werden. Jede Ausnahme enthält die spezifische Fehlermeldung, gefolgt von einer kurzen Beschreibung und Vorschlägen für mögliche Lösungen.

AccessDeniedException

Meldung: Benutzer: *`userARN`* ist nicht berechtigt, Folgendes auszuführen: iam:PassRole on-Ressource: *`roleARN`*

Sie sind nicht berechtigt, eine IAM-Rolle an den QLDB-Service zu übergeben. QLDB benötigt eine Rolle für alle Journalexportanforderungen, und Sie müssen über Berechtigungen verfügen, um diese Rolle an QLDB zu übergeben. Die Rolle stellt QLDB Schreibberechtigungen in Ihrem angegebenen Amazon S3-Bucket bereit.

Stellen Sie sicher, dass Sie eine IAM-Richtlinie definieren, die die Berechtigung zum Ausführen der `PassRole` API-Operation für die von Ihnen angegebene IAM-Rollenressource für den QLDB-Service (`qldb.amazonaws.com`) gewährt. Eine Beispielrichtlinie finden Sie in Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

IllegalArgumentException

Meldung: QLDB encountered an error validating S3 configuration: *`errorCode errorMessage`*

Eine mögliche Ursache für diesen Fehler ist, dass der bereitgestellte Amazon S3-Bucket in Amazon S3 nicht vorhanden ist. Oder QLDB verfügt nicht über genügend Berechtigungen, um Objekte in den angegebenen Amazon S3-Bucket zu schreiben.

Stellen Sie sicher, dass der S3-Bucket-Name, den Sie in Ihrer Anforderung für den Exportauftrag angeben, korrekt ist. Weitere Informationen zur Bucket-Benennung finden Sie unter Bucket-Einschränkungen und -Einschränkungen im Benutzerhandbuch für Amazon Simple Storage Service.

Stellen Sie außerdem sicher, dass Sie eine Richtlinie für den angegebenen Bucket definieren, die dem QLDB-Service () - `PutObject` und -`PutObjectAcl`Berechtigungen gewährt`qldb.amazonaws.com`. Weitere Informationen hierzu finden Sie unter Exportberechtigungen.

IllegalArgumentException

Meldung: Unerwartete Antwort von Amazon S3 bei der Validierung der S3-Konfiguration. Antwort von S3: *`errorCode errorMessage`*

Der Versuch, Journalexportdaten in den bereitgestellten S3-Bucket zu schreiben, ist mit der bereitgestellten Amazon S3-Fehlerantwort fehlgeschlagen. Weitere Informationen zu möglichen Ursachen finden Sie unter Fehlerbehebung bei Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.

IllegalArgumentException

Meldung: Amazon S3 bucket prefix must not exceed 128 characters

Das in der Journal-Journalexportanfrage angegebene Präfix enthält mehr als 128 Zeichen.

IllegalArgumentException

Meldung: Start date must not be greater than end date

Sowohl `InclusiveStartTime` als auch `ExclusiveEndTime` müssen im Datums- und Zeitformat ISO 8601 und in koordinierter Weltzeit (Coordinated Universal Time, UTC) angegeben werden.

IllegalArgumentException

Meldung: End date cannot be in the future

Sowohl `InclusiveStartTime` als auch `ExclusiveEndTime` müssen im Datums- und Zeitformat `ISO 8601` und in koordinierter Weltzeit (Coordinated Universal Time, UTC) angegeben werden.

IllegalArgumentException

Meldung: Die angegebene Objektverschlüsselungseinstellung (S3EncryptionConfiguration) ist nicht mit einem AWS Key Management Service (AWS KMS)-Schlüssel kompatibel

Sie haben einen `KMSKeyArn` mit einem `ObjectEncryptionType` vom entweder `NO_ENCRYPTION` oder `SSE_S3` angegeben. Sie können nur einen vom Kunden verwalteten AWS KMS key für einen Objektverschlüsselungstyp von angeben`SSE_KMS`. Weitere Informationen zu serverseitigen Verschlüsselungsoptionen in Amazon S3 finden Sie unter Schützen von Daten mit serverseitiger Verschlüsselung im Amazon-S3-EntwicklerhandbuchAmazon S3.

LimitExceededException

Meldung: Exceeded the limit of 2 concurrently running Journal export jobs

QLDB erzwingt ein Standardlimit von zwei gleichzeitigen Journalexportaufträgen.

Streaming von Journaldaten aus Amazon QLDB

Amazon QLDB verwendet ein unveränderliches Transaktionsprotokoll, das als Journal bezeichnet wird, für die Datenspeicherung. Das Journal verfolgt jede Änderung an Ihren zugesagten Daten und führt einen vollständigen und überprüfbaren Verlauf der Änderungen im Laufe der Zeit.

Sie können einen Stream in QLDB erstellen, der jede Dokumentrevision erfasst, die an Ihr Journal übergeben wird, und diese Daten nahezu in Echtzeit an Amazon Kinesis Data Streams liefert. Ein QLDB-Stream ist ein kontinuierlicher Datenfluss aus dem Ledger-Journal zu einer Kinesis-Datenstromressource.

Anschließend verwenden Sie die Kinesis-Streaming-Plattform oder die Kinesis Client Library, um Ihren Stream zu verbrauchen, die Datensätze zu verarbeiten und den Dateninhalt zu analysieren. Ein QLDB-Stream schreibt Ihre Daten in Kinesis Data Streams in drei Arten von Datensätzen: Kontrolle , Blockzusammenfassung und Revisionsdetails . Weitere Informationen finden Sie unter QLDB-Stream-Datensätze in Kinesis.

Themen

Häufige Anwendungsfälle

Mit Streaming können Sie QLDB als eine einzige, überprüfbare Informationsquelle verwenden und gleichzeitig Ihre Journaldaten in andere -Services integrieren. Im Folgenden sind einige der häufigsten Anwendungsfälle aufgeführt, die von QLDB-Journalstreams unterstützt werden:

- Ereignisgesteuerte Architektur – Erstellen Sie Anwendungen in einem ereignisgesteuerten Architekturstil mit entkoppelten Komponenten. Beispielsweise kann eine Bank AWS Lambda Funktionen verwenden, um ein Benachrichtigungssystem zu implementieren, das Kunden benachrichtigt, wenn ihr Kontosaldo unter einen Schwellenwert fällt. In einem solchen System werden die Kontosalden in einem QLDB-Ledger verwaltet und alle Saldoänderungen werden im Journal aufgezeichnet. Die AWS Lambda Funktion kann die Benachrichtigungslogik auslösen, wenn sie ein Saldoaktualisierungsereignis verbraucht, das an das Journal übergeben und an einen Kinesis-Datenstrom gesendet wird.
- Echtzeitanalysen – Erstellen Sie Kinesis-Konsumentenanwendungen, die Echtzeitanalysen von Ereignisdaten ausführen. Mit dieser Funktion können Sie nahezu in Echtzeit Einblicke gewinnen und schnell auf eine sich verändernde Geschäftsumgebung reagieren. Beispielsweise kann eine E-Commerce-Website Produktverkaufsdaten analysieren und Werbung für einen reduzierten Artikel stoppen, sobald der Umsatz ein Limit erreicht.
- Historische Analysen – Nutzen Sie die journalorientierte Architektur von Amazon QLDB, indem Sie historische Ereignisdaten wiederholen. Sie können einen QLDB-Stream ab einem beliebigen Zeitpunkt in der Vergangenheit starten, an dem alle Revisionen seit diesem Zeitpunkt an Kinesis Data Streams übermittelt werden. Mit dieser Funktion können Sie Kinesis-Konsumentenanwendungen erstellen, die Analyseaufträge für historische Daten ausführen. Beispielsweise kann eine E-Commerce-Website nach Bedarf Analysen ausführen, um frühere Verkaufsmetriken zu generieren, die zuvor nicht erfasst wurden.
- Replikation in speziell entwickelte Datenbanken – Verbinden Sie QLDB-Ledger mit anderen speziell entwickelten Datenspeichern mithilfe von QLDB-Journalstreams. Verwenden Sie beispielsweise die Kinesis-Streaming-Datenplattform, um in Amazon OpenSearch Service zu integrieren, der Volltextsuchfunktionen für QLDB-Dokumente bereitstellen kann. Sie können auch benutzerdefinierte Kinesis-Konsumentenanwendungen erstellen, um Ihre Journaldaten in andere speziell entwickelte Datenbanken zu replizieren, die unterschiedliche materialisierte Ansichten bieten. Replizieren Sie beispielsweise für relationale Daten in Amazon Aurora oder für diagrammbasierte Daten in Amazon Neptune.

Nutzen Ihres Streams

Verwenden Sie Kinesis Data Streams, um große Streams von Datensätzen kontinuierlich zu verbrauchen, zu verarbeiten und zu analysieren. Zusätzlich zu Kinesis Data Streams umfasst die Kinesis-Streaming-Datenplattform Amazon Data Firehose und Amazon Managed Service für Apache Flink. Sie können diese Plattform verwenden, um Datensätze direkt an Services wie Amazon OpenSearch Service, Amazon Redshift, Amazon S3 oder Splunk zu senden. Weitere Informationen

finden Sie unter Verbraucher von Kinesis Data Streams im Entwicklerhandbuch für Amazon Kinesis Data Streams.

Sie können auch die Kinesis Client Library (KCL) verwenden, um eine Stream-Konsumentenanwendung zur benutzerdefinierten Verarbeitung von Datensätzen zu erstellen. Die KCL vereinfacht die Codierung durch Bereitstellen nützlicher Abstraktionen oberhalb der Low-Level-Kinesis-Data-Streams-API. Weitere Informationen zur KCL finden Sie unter Verwenden der Kinesis Client Library im Entwicklerhandbuch für Amazon Kinesis Data Streams.

Liefergarantie

QLDB-Streams bieten eine at-least-once Liefergarantie. Jeder Datensatz, der von einem QLDB-Stream erstellt wird, wird mindestens einmal an Kinesis Data Streams übermittelt. Dieselben Datensätze können mehrmals in einem Kinesis-Datenstrom angezeigt werden. Es muss also eine Deduplizierungslogik in der Verbraucheranwendungsebene vorhanden sein, wenn Ihr Anwendungsfall dies erfordert.

Es gibt auch keine Bestellgarantien. Unter bestimmten Umständen können QLDB-Blöcke und -Revisionen in einem Kinesis-Datenstrom außerhalb der Reihenfolge erstellt werden. Weitere Informationen finden Sie unter Umgang mit Duplikaten und out-of-order Datensätzen.

Überlegungen zur Bereitstellungslatenz

QLDB-Streams stellen Updates für Kinesis Data Streams in der Regel nahezu in Echtzeit bereit. In den folgenden Szenarien kann es jedoch zu einer zusätzlichen Latenz kommen, bevor neu festgeschriebene QLDB-Daten an einen Kinesis-Datenstrom ausgegeben werden:

- Kinesis kann Daten drosseln, die von QLDB gestreamt werden, abhängig von Ihrer Bereitstellung von Kinesis Data Streams. Dies kann beispielsweise der Fall sein, wenn Sie mehrere QLDB-Streams haben, die in einen einzelnen Kinesis-Datenstrom schreiben, und die Anforderungsrate von QLDB die Kapazität der Kinesis-Stream-Ressource überschreitet. Die Drosselung in Kinesis kann auch bei Verwendung der On-Demand-Bereitstellung auftreten, wenn der Durchsatz in weniger als 15 Minuten auf mehr als das Doppelte des vorherigen Spitzenwerts anwächst.

  Sie können diesen überschrittenen Durchsatz messen, indem Sie die Kinesis-Metrik überwachen`WriteProvisionedThroughputExceeded`. Weitere Informationen und mögliche Lösungen finden Sie unter Wie behebe ich Drosselungsfehler in Kinesis Data Streams?.

- Mit QLDB-Streams können Sie einen unbegrenzten Stream mit einem Startdatum und einer Startzeit in der Vergangenheit und ohne Enddatum und -zeit erstellen. QLDB beginnt standardmäßig erst dann neu festgeschriebene Daten an Kinesis Data Streams auszugeben, nachdem alle vorherigen Daten aus dem angegebenen Startdatum und der angegebenen Startzeit erfolgreich übermittelt wurden. Wenn Sie in diesem Szenario eine zusätzliche Latenz feststellen, müssen Sie möglicherweise warten, bis die vorherigen Daten zugestellt werden, oder Sie können den Stream ab einem späteren Startdatum und einer späteren Startzeit starten.

Erste Schritte mit Streams

Im Folgenden finden Sie einen allgemeinen Überblick über die Schritte, die erforderlich sind, um mit dem Streamen von Journaldaten an Kinesis Data Streams zu beginnen:

1. Erstellen Sie eine Kinesis Data Streams-Ressource. Anweisungen finden Sie unter Erstellen und Aktualisieren von Datenströmen im Entwicklerhandbuch für Amazon Kinesis Data Streams.
2. Erstellen Sie eine IAM-Rolle, die es QLDB ermöglicht, Schreibberechtigungen für den Kinesis-Datenstrom zu übernehmen. Anweisungen finden Sie unter Stream-Berechtigungen in QLDB.
3. Erstellen Sie einen QLDB-Journalstream. Anweisungen finden Sie unter Erstellen und Verwalten von Streams in QLDB.
4. Nutzen Sie den Kinesis-Datenstrom, wie im vorherigen Abschnitt beschriebenNutzen Ihres Streams. Codebeispiele, die zeigen, wie die Kinesis Client Library oder verwendet wird AWS Lambda, finden Sie unter Entwickeln mit Streams in QLDB.

Erstellen und Verwalten von Streams in QLDB

Amazon QLDB bietet API-Operationen zum Erstellen und Verwalten eines Streams von Journaldaten aus Ihrem Ledger an Amazon Kinesis Data Streams. Der QLDB-Stream erfasst jede Dokumentrevision, die an Ihr Journal übergeben wird, und sendet sie an einen Kinesis-Datenstrom.

Sie können die AWS Management Console, ein AWS SDK oder die AWS Command Line Interface (AWS CLI) verwenden, um einen Journalstream zu erstellen. Darüber hinaus können Sie auch eine -AWS CloudFormationVorlage verwenden, um Streams zu erstellen. Weitere Informationen finden Sie in der -AWS::QLDB::StreamRessource im AWS CloudFormation -Benutzerhandbuch.

Themen

- Stream-Parameter

- Stream-ARN
- AWS Management Console
- Stream-Zustände
- Umgang mit beeinträchtigten Streams

Stream-Parameter

Um einen QLDB-Journalstream zu erstellen, müssen Sie die folgenden Konfigurationsparameter angeben:

Ledger-Name

Das QLDB-Ledger, dessen Journaldaten Sie an Kinesis Data Streams streamen möchten.

Stream-Name

Der Name, den Sie dem QLDB-Journal-Stream zuweisen möchten. Benutzerdefinierte Namen können helfen, den Zweck eines Streams zu identifizieren und anzugeben.

Der Stream-Name muss unter anderen aktiven Streams für ein bestimmtes Ledger eindeutig sein. Stream-Namen haben die gleichen in Kontingente und Limits in Amazon QLDB definierten Benennungsbeschränkungen wie Ledger-Namen.

Zusätzlich zum Stream-Namen weist QLDB jedem von Ihnen erstellten QLDB-Stream eine Stream-ID zu. Die Stream-ID ist unabhängig von ihrem Status unter allen Streams für ein bestimmtes Ledger eindeutig.

Startdatum und -zeit

Datums- und Uhrzeitangaben zum Zeitpunkt, ab dem das Streamen von Journaldaten gestartet werden soll. Dieser Wert kann ein beliebiges Datum und eine beliebige Uhrzeit in der Vergangenheit, aber nicht in der Zukunft liegen.

Enddatum und -zeit

(Optional) Das Datum und die Uhrzeit, die angeben, wann der Stream endet.

Wenn Sie einen unbefristeten Stream ohne Endzeit erstellen, müssen Sie ihn manuell abbrechen, um den Stream zu beenden. Sie können auch einen aktiven, endlichen Stream abbrechen, der das angegebene Enddatum und die angegebene Uhrzeit noch nicht erreicht hat.

Ziel-Kinesis-Datenstrom

Die Zielressource von Kinesis Data Streams, in die Ihr Stream die Datensätze schreibt. Informationen zum Erstellen eines Kinesis-Datenstroms finden Sie unter Erstellen und Aktualisieren von Datenströmen im Entwicklerhandbuch für Amazon Kinesis Data Streams.

> **Important**
>
> - Regionsübergreifende und kontoübergreifende Streams werden nicht unterstützt. Der angegebene Kinesis-Datenstrom muss sich im selben AWS-Region und Konto wie Ihr Ledger befinden.
> - Die Datensatzaggregation in Kinesis Data Streams ist standardmäßig aktiviert. Mit dieser Option kann QLDB mehrere Datensätze in einem einzigen Datensatz von Kinesis Data Streams veröffentlichen, wodurch die Anzahl der pro API-Aufruf gesendeten Datensätze erhöht wird.
>
>   Die Datensatzaggregation hat wichtige Auswirkungen auf die Verarbeitung von Datensätzen und erfordert eine Deaggregation in Ihrem Stream-Verbraucher. Weitere Informationen finden Sie unter KPL-Schlüsselkonzepte und Verbraucher-Deaggregation im Entwicklerhandbuch für Amazon Kinesis Data Streams.

IAM-Rolle

Die IAM-Rolle, die es QLDB ermöglicht, Schreibberechtigungen für Ihren Kinesis-Datenstrom zu übernehmen. Sie können die QLDB-Konsole verwenden, um diese Rolle automatisch zu erstellen, oder Sie können sie manuell in IAM erstellen. Informationen zum manuellen Erstellen finden Sie unter Stream-Berechtigungen.

Um beim Anfordern eines Journalstreams eine Rolle an QLDB zu übergeben, müssen Sie über Berechtigungen zum Ausführen der `iam:PassRole`-Aktion für die IAM-Rollenressource verfügen.

Stream-ARN

Jeder QLDB-Journalstream ist eine Unterressource eines Ledgers und wird eindeutig durch einen Amazon-Ressourcennamen (ARN) identifiziert. Im Folgenden finden Sie ein Beispiel-ARN eines QLDB-Streams mit der Stream-ID `IiPT4brpZCqCq3f4MTHbYy` für einen Ledger mit dem Namen `exampleLedger`.

```
arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/IiPT4brpZCqCq3f4MTHbYy
```

Im folgenden Abschnitt wird beschrieben, wie Sie einen QLDB-Stream mithilfe der erstellen und abbrechen AWS Management Console.

AWS Management Console

Gehen Sie wie folgt vor, um einen QLDB-Stream mit der QLDB-Konsole zu erstellen oder abzubrechen.

So erstellen Sie einen Stream (Konsole)

1. Melden Sie sich bei der an AWS Management Consoleund öffnen Sie die Amazon-QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Klicken Sie im Navigationsbereich auf Streams.
3. Wählen Sie QLDB-Stream erstellen aus.
4. Geben Sie auf der Seite QLDB-Stream erstellen die folgenden Einstellungen ein:

   - Stream-Name – Der Name, den Sie dem QLDB-Stream zuweisen möchten.
   - Ledger – Der Ledger, dessen Journaldaten Sie streamen möchten.
   - Startdatum und -zeit – Der inklusive Zeitstempel in koordinierter Weltzeit (UTC), ab dem mit dem Streamen von Journaldaten begonnen werden soll. Dieser Zeitstempel zeigt standardmäßig das aktuelle Datum und die aktuelle Uhrzeit. Er darf nicht in der Zukunft liegen und muss vor dem Enddatum und der Endzeit liegen.
   - Enddatum und -zeit – (Optional) Der exklusive Zeitstempel (UTC), der angibt, wann der Stream endet. Wenn Sie diesen Parameter leer lassen, wird der Stream auf unbestimmte Zeit ausgeführt, bis Sie ihn abbrechen.
   - Zielstream – Die Zielressource von Kinesis Data Streams, in die Ihr Stream die Datensätze schreibt. Verwenden Sie das folgende ARN-Format.

     ```
     arn:aws:kinesis:aws-region:account-id:stream/kinesis-stream-name
     ```

     Im Folgenden wird ein Beispiel gezeigt.

     ```
     arn:aws:kinesis:us-east-1:123456789012:stream/stream-for-qldb
     ```

Regionsübergreifende und kontoübergreifende Streams werden nicht unterstützt. Der angegebene Kinesis-Datenstrom muss sich im selben AWS-Region und Konto wie Ihr Ledger befinden.

- Aktivieren der Datensatzaggregation in Kinesis Data Streams – (Standardmäßig aktiviert) Ermöglicht QLDB die Veröffentlichung mehrerer Datensätze in einem einzigen Datensatz von Kinesis Data Streams, wodurch die Anzahl der pro API-Aufruf gesendeten Datensätze erhöht wird.
- Servicezugriff – Die IAM-Rolle, die QLDB Schreibberechtigungen für Ihren Kinesis-Datenstrom gewährt.

  Um beim Anfordern eines Journalstreams eine Rolle an QLDB zu übergeben, müssen Sie über Berechtigungen zum Ausführen der *`iam:PassRole`* Aktion für die IAM-Rollenressource verfügen.

  - Erstellen und Verwenden einer neuen Servicerolle – Lassen Sie die Konsole eine neue Rolle für Sie mit den erforderlichen Berechtigungen für den angegebenen Kinesis-Datenstrom erstellen.
  - Verwenden einer vorhandenen Servicerolle – Informationen zum manuellen Erstellen dieser Rolle in IAM finden Sie unter Stream-Berechtigungen.
- Tags – (Optional) Fügen Sie dem Stream Metadaten hinzu, indem Sie Tags als Schlüssel-Wert-Paare anfügen. Sie können Ihrem Stream Tags hinzufügen, um sie zu organisieren und zu identifizieren. Weitere Informationen finden Sie unter Markieren von Amazon-QLDB Ressourcen.

  Wählen Sie Add tag (Tag hinzufügen) aus und geben Sie dann beliebige Schlüssel-Wert-Paare ein.

5. Wenn Sie die gewünschten Einstellungen vorgenommen haben, wählen Sie QLDB-Stream erstellen aus.

   Wenn Ihre Anforderung erfolgreich übermittelt wurde, kehrt die Konsole zur Hauptseite Streams zurück und listet Ihre QLDB-Streams mit ihrem aktuellen Status auf.

6. Nachdem Ihr Stream aktiv ist, verwenden Sie Kinesis, um Ihre Stream-Daten mit einer Konsumentenanwendung zu verarbeiten.

   Öffnen Sie die Kinesis-Data-Streams-Konsole unter https://console.aws.amazon.com/kinesis/.

Hinweise zum Format der Stream-Datensätze finden Sie unter QLDB-Stream-Datensätze in Kinesis.

Informationen zum Umgang mit Streams, die zu einem Fehler führen, finden Sie unter Umgang mit beeinträchtigten Streams.

So brechen Sie einen Stream ab (Konsole)

Sie können einen QLDB-Stream nicht neu starten, nachdem Sie ihn abgebrochen haben. Um die Bereitstellung Ihrer Daten an Kinesis Data Streams fortzusetzen, können Sie einen neuen QLDB-Stream erstellen.

1. Öffnen Sie die Amazon-QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Klicken Sie im Navigationsbereich auf Streams.
3. Wählen Sie in der Liste der QLDB-Streams den aktiven Stream aus, den Sie abbrechen möchten.
4. Wählen Sie Cancel stream (Stream abbrechen) aus. Bestätigen Sie dies, indem Sie **`cancel stream`** in das bereitgestellte Feld eingeben.

Informationen zur Verwendung der QLDB-API mit einem AWS -SDK oder der AWS CLI zum Erstellen und Verwalten von Journalstreams finden Sie unter Entwickeln mit Streams in QLDB.

Stream-Zustände

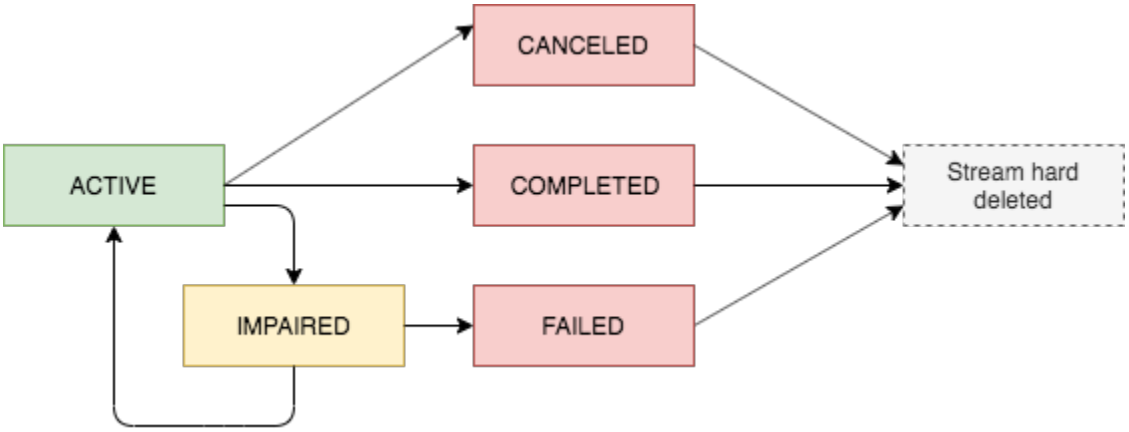
Der Status eines QLDB-Streams kann einer der folgenden sein:

- `ACTIVE` – Streamt derzeit Daten oder wartet darauf, Daten zu streamen (für einen unbegrenzten Stream ohne Endzeit).
- `COMPLETED` – Hat das Streaming aller Journalblöcke innerhalb des angegebenen Zeitraums erfolgreich abgeschlossen. Diese ist ein Terminalstatus.
- `CANCELED` – wurde von einer Benutzeranfrage vor der angegebenen Endzeit beendet und streamt keine Daten mehr. Diese ist ein Terminalstatus.
- `IMPAIRED` – kann aufgrund eines Fehlers, der Ihre Aktion erfordert, keine Datensätze in Kinesis schreiben. Dies ist ein wiederherstellbarer, nicht beendender Zustand.

  Wenn Sie den Fehler innerhalb einer Stunde beheben, wechselt der Stream automatisch in den `ACTIVE`-Zustand. Wenn der Fehler nach einer Stunde ungelöst bleibt, wechselt der Stream automatisch in den `FAILED`-Zustand.

- `FAILED` – kann aufgrund eines Fehlers keine Datensätze in Kinesis schreiben und befindet sich in einem nicht behebbaren Endzustand.

Das folgende Diagramm zeigt, wie eine QLDB-Stream-Ressource zwischen Zuständen wechseln kann.



Ablaufdatum für Beendigungs-Streams

Stream-Ressourcen, die sich in einem Beendigungszustand (`CANCELED`, `COMPLETED`, und `FAILED`) befinden, unterliegen einem Aufbewahrungszeitraum von 7 Tagen. Sie werden nach Ablauf dieses Limits automatisch dauerhaft gelöscht.

Nachdem ein Terminal-Stream gelöscht wurde, können Sie die QLDB-Konsole oder die QLDB-API nicht mehr verwenden, um die Stream-Ressource zu beschreiben oder aufzulisten.

Umgang mit beeinträchtigten Streams

Wenn in Ihrem Stream ein Fehler auftritt, wechselt er zuerst in den `IMPAIRED` Status . QLDB wiederholt `IMPAIRED` Streams bis zu eine Stunde lang.

Wenn Sie den Fehler innerhalb einer Stunde beheben, wechselt der Stream automatisch in den `ACTIVE`-Zustand. Wenn der Fehler nach einer Stunde ungelöst bleibt, wechselt der Stream automatisch in den `FAILED`-Zustand.

Ein beeinträchtigter oder fehlgeschlagener Stream kann eine der folgenden Fehlerursachen haben:

- `KINESIS_STREAM_NOT_FOUND` – Die Zielressource von Kinesis Data Streams ist nicht vorhanden. Stellen Sie sicher, dass der Kinesis-Datenstrom, den Sie in Ihrer QLDB-Stream-Anforderung angegeben haben, korrekt ist. Gehen Sie dann zu Kinesis und erstellen Sie den von Ihnen angegebenen Datenstrom.

- `IAM_PERMISSION_REVOKED` – QLDB verfügt nicht über genügend Berechtigungen, um Datensätze in den angegebenen Kinesis-Datenstrom zu schreiben. Stellen Sie sicher, dass Sie eine Richtlinie für den angegebenen Kinesis-Datenstrom definieren, die dem QLDB-Service (`qldb.amazonaws.com`) Berechtigungen für die folgenden Aktionen gewährt:
  - `kinesis:PutRecord`
  - `kinesis:PutRecords`
  - `kinesis:DescribeStream`
  - `kinesis:ListShards`

Überwachung beeinträchtigter Streams

Wenn ein Stream beeinträchtigt wird, zeigt die QLDB-Konsole ein Banner an, das Details zum Stream und den aufgetretenen Fehler anzeigt. Sie können auch die `DescribeJournalKinesisStream` -API-Operation verwenden, um den Status eines Streams und die zugrunde liegende Fehlerursache abzurufen.

Darüber hinaus können Sie Amazon verwenden, CloudWatch um einen Alarm zu erstellen, der die `IsImpaired` Metrik eines Streams überwacht. Informationen zur Überwachung von QLDB-Metriken mit CloudWatchfinden Sie unter Amazon-QLDB-Dimensionen und -Metriken.

Entwickeln mit Streams in QLDB

In diesem Abschnitt werden die API-Operationen zusammengefasst, die Sie mit einem AWS -SDK oder verwenden können AWS CLI , um Journalstreams in Amazon QLDB zu erstellen und zu verwalten. Außerdem werden die Beispielanwendungen beschrieben, die diese Operationen demonstrieren und die Kinesis Client Library (KCL) oder verwenden AWS Lambda , um einen Stream-Verbraucher zu implementieren.

Sie können die KCL verwenden, um Konsumentenanwendungen für Amazon Kinesis Data Streams zu erstellen. Die KCL vereinfacht die Codierung durch Bereitstellen nützlicher Abstraktionen oberhalb der Low-Level-Kinesis-Data-Streams-API. Weitere Informationen zur KCL finden Sie unter Verwenden der Kinesis Client Library im Entwicklerhandbuch für Amazon Kinesis Data Streams.

Inhalt

- Grundlegende Operationen (Java)
- Integration mit OpenSearch Service (Python)
- Integration mit Amazon SNS und Amazon SQS (Python)

QLDB-Journal-Stream-APIs

Die QLDB-API bietet die folgenden Journalstream-Operationen zur Verwendung durch Anwendungsprogramme:

- `StreamJournalToKinesis` – Erstellt einen Journalstream für einen bestimmten QLDB-Ledger. Der Stream erfasst jede Dokumentrevision, die an das Journal des Ledgers übergeben wird, und stellt die Daten an eine angegebene Kinesis Data Streams-Ressource bereit.
  - Die Datensatzaggregation in Kinesis Data Streams ist standardmäßig aktiviert. Mit dieser Option kann QLDB mehrere Datensätze in einem einzigen Datensatz von Kinesis Data Streams veröffentlichen, wodurch die Anzahl der pro API-Aufruf gesendeten Datensätze erhöht wird.

    Die Datensatzaggregation hat wichtige Auswirkungen auf die Verarbeitung von Datensätzen und erfordert eine Deaggregation in Ihrem Stream-Verbraucher. Weitere Informationen finden Sie unter KPL-Schlüsselkonzepte und Verbraucher-Deaggregation im Entwicklerhandbuch für Amazon Kinesis Data Streams.
- `DescribeJournalKinesisStream` – Gibt detaillierte Informationen zu einem bestimmten QLDB-Journal-Stream zurück. Die Ausgabe enthält den ARN, den Stream-Namen, den aktuellen Status, die Erstellungszeit und die Parameter Ihrer ursprünglichen Stream-Erstellungsanforderung.
- `ListJournalKinesisStreamsForLedger` – Gibt eine Liste aller QLDB-Journal-Stream-Deskriptoren für einen bestimmten Ledger zurück. Die Ausgabe jedes Stream-Deskriptors enthält die gleichen Details, die von `DescribeJournalKinesisStream` zurückgegeben werden.
- `CancelJournalKinesisStream` – Beendet einen bestimmten QLDB-Journal-Stream. Bevor ein Stream abgebrochen werden kann, muss sein aktueller Status „`ACTIVE`“ lauten.

  Sie können einen Stream nicht neu starten, nachdem Sie ihn abgebrochen haben. Um die Bereitstellung Ihrer Daten an Kinesis Data Streams fortzusetzen, können Sie einen neuen QLDB-Stream erstellen.

Eine vollständige Beschreibung dieser API-Vorgänge finden Sie unter Amazon QLDB API-Referenz.

Informationen zum Erstellen und Verwalten von Journalstreams mit der finden Sie AWS CLIin der AWS CLI -Befehlsreferenz.

Beispielanwendungen

QLDB bietet Beispielanwendungen, die verschiedene Operationen mithilfe von Journalstreams demonstrieren. Diese Anwendungen sind Open Source auf der AWS GitHub Website Samples .

Themen

- Grundlegende Operationen (Java)
- Integration mit OpenSearch Service (Python)
- Integration mit Amazon SNS und Amazon SQS (Python)

Grundlegende Operationen (Java)

Ein Java-Codebeispiel, das grundlegende Operationen für QLDB-Journalstreams demonstriert, finden Sie im GitHub Repository aws-samples/amazon-qldb-dmv-sample-java . Anweisungen zum Herunterladen und Installieren dieser Beispielanwendung finden Sie unter Installation der Amazon QLDB Java-Beispielanwendung.

> **Note**
>
> Fahren Sie nach der Installation der Anwendung nicht mit Schritt 1 des Java-Tutorials fort, um einen Ledger zu erstellen. Diese Beispielanwendung für Streaming erstellt das `vehicle-registration`-Ledger für Sie.

Diese Beispielanwendung verpackt den vollständigen Quellcode aus Java-Anleitung und seinen Abhängigkeiten, einschließlich der folgenden Module:

- AWS SDK for Java – Zum Erstellen und Löschen der QLDB- und Kinesis Data Streams-Ressourcen, einschließlich Ledger, QLDB-Journalstreams und Kinesis-Datenströmen.
- Amazon QLDB-Treiber für Java – Zum Ausführen von Datentransaktionen auf einem Ledger mithilfe von PartiQL-Anweisungen, einschließlich des Erstellens von Tabellen und Einfügens von Dokumenten.
- Kinesis Client Library – Zum Verarbeiten von Daten aus einem Kinesis-Datenstrom.

Ausführen des Codes

Die StreamJournal-Klasse enthält Tutorial-Code, der die folgenden Vorgänge veranschaulicht:

1. Erstellen Sie ein Ledger mit dem Namen „`vehicle-registration`“, erstellen Sie Tabellen, und laden Sie sie mit Beispieldaten.

   > **Note**
   >
   > Bevor Sie diesen Code ausführen, stellen Sie sicher, dass noch kein aktives Ledger mit dem Namen „`vehicle-registration`“ vorhanden ist.

2. Erstellen Sie einen Kinesis-Datenstrom, eine IAM-Rolle, mit der QLDB Schreibberechtigungen für den Kinesis-Datenstrom übernehmen kann, und einen QLDB-Journalstream.
3. Verwenden Sie die KCL, um einen Stream-Reader zu starten, der den Kinesis-Datenstrom verarbeitet und jeden QLDB-Datensatz protokolliert.
4. Verwenden Sie die Stream-Daten, um die Hashkette des `vehicle-registration`-Beispiel-Ledgers zu überprüfen.
5. Bereinigen Sie alle Ressourcen, indem Sie den Stream-Reader stoppen, den QLDB-Journal-Stream abbrechen, den Ledger löschen und den Kinesis-Datenstrom löschen.

Um den `StreamJournal`-Tutorial-Code auszuführen, geben Sie den folgenden Gradle-Befehl aus Ihrem Projekt-Stammverzeichnis ein.

```
./gradlew run -Dtutorial=streams.StreamJournal
```

Integration mit OpenSearch Service (Python)

Eine Python-Beispielanwendung, die zeigt, wie ein QLDB-Stream in Amazon OpenSearch Service integriert wird, finden Sie im GitHub Repository aws-samples/amazon-qldb-streaming-amazon-opensearch-service-sample-python. Diese Anwendung verwendet eine - AWS Lambda Funktion, um einen Kinesis-Data-Streams-Verbraucher zu implementieren.

Geben Sie den folgenden `git`-Befehl ein, um das Repository zu klonen.

```
git clone https://github.com/aws-samples/amazon-qldb-streaming-amazon-opensearch-service-sample-python.git
```

Anweisungen zum Ausführen der Beispielanwendung finden Sie unter README auf GitHub .

Integration mit Amazon SNS und Amazon SQS (Python)

Eine Python-Beispielanwendung, die zeigt, wie ein QLDB-Stream in Amazon Simple Notification Service (Amazon SNS) integriert wird, finden Sie im GitHub Repository aws-samples/amazon-qldb-streams-dmv-sample-lambda-python.

Diese Anwendung verwendet eine - AWS Lambda Funktion, um einen Kinesis-Data-Streams-Verbraucher zu implementieren. Es sendet Nachrichten an ein Amazon SNS-Thema, das eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange abonniert hat.

Geben Sie den folgenden `git`-Befehl ein, um das Repository zu klonen.

```
git clone https://github.com/aws-samples/amazon-qldb-streams-dmv-sample-lambda-
python.git
```

Anweisungen zum Ausführen der Beispielanwendung finden Sie unter README auf GitHub .

QLDB-Stream-Datensätze in Kinesis

Ein Amazon-QLDB-Stream schreibt drei Arten von Datensätzen in eine bestimmte Amazon Kinesis-Data-Streams-Ressource: Kontrolle , Blockübersicht und Revisionsdetails . Alle drei Datensatztypen werden in die binäre Darstellung des Amazon-Ion-Formats geschrieben.

Kontrolldatensätze geben den Start und den Abschluss Ihrer QLDB-Streams an. Wenn eine Revision in Ihr Journal übernommen wird, schreibt ein QLDB-Stream alle zugehörigen Journalblockdaten in Datensätze mit Blockübersichts- und Revisionsdetails.

Die drei Datensatztypen sind polymorph. Sie bestehen alle aus einem gemeinsamen Datensatz der obersten Ebene, der den QLDB-Stream-ARN, den Datensatztyp und die Datensatznutzlast enthält. Dieser Datensatz der obersten Ebene hat das folgende Format.

```
{
  qldbStreamArn: string,
  recordType: string,
  payload: {
    //control | block summary | revision details record
  }
}
```

Das `recordType`-Feld kann einen von drei Werten haben:

- `CONTROL`
- `BLOCK_SUMMARY`
- `REVISION_DETAILS`

In den folgenden Abschnitten wird das Format und der Inhalt jedes einzelnen Nutzlastdatensatzes beschrieben.

> **Note**
>
> QLDB schreibt alle Stream-Datensätze in Kinesis Data Streams in der binären Darstellung von Amazon Ion. Die folgenden Beispiele sind in der Textdarstellung von Ion enthalten, um den Datensatzinhalt in einem lesbaren Format zu veranschaulichen.

Themen

Datensätze steuern

Ein QLDB-Stream schreibt Kontrolldatensätze, um seine Start- und Abschlussereignisse anzugeben. Nachfolgend finden Sie Beispiele für Steuerdatensätze mit Beispieldaten für jeden `controlRecordType`:

- `CREATED` – Der erste Datensatz, den ein QLDB-Stream in Kinesis schreibt, um anzuzeigen, dass Ihr neu erstellter Stream aktiv ist.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
    controlRecordType:"CREATED"
```

```
  }
}
```

- `COMPLETED` – Der letzte Datensatz, den ein QLDB-Stream in Kinesis schreibt, um anzuzeigen, dass Ihr Stream das angegebene Enddatum und die angegebene Endzeit erreicht hat. Dieser Datensatz wird nicht geschrieben, wenn Sie den Stream abbrechen.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
    controlRecordType:"COMPLETED"
  }
}
```

Blockzusammenfassungsdatensätze

Ein Blockzusammenfassungsdatensatz stellt einen Journalblock dar, in dem die Dokumentversionen festgeschrieben werden. Ein Block ist ein Objekt, das während einer Transaktion an Ihr QLDB-Journal übergeben wird.

Die Nutzlast eines Blockzusammenfassungsdatensatzes enthält die Blockadresse, den Zeitstempel und andere Metadaten der Transaktion, die der Block festgeschrieben hat. Sie enthält auch zusammenfassende Attribute der Revisionen im Block und die PartiQL-Anweisungen, die sie festgeschrieben haben. Im Folgenden finden Sie ein Beispiel für einen Blockzusammenfassungsdatensatz mit Beispieldaten.

> **Note**
>
> Dieses Blockübersichtsbeispiel dient nur zu Informationszwecken. Die angezeigten Hashwerte sind keine tatsächlich berechneten Hashwerte.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"BLOCK_SUMMARY",
  payload:{
```

```
    blockAddress:{
      strandId:"ElYL30RGoqrFCbbaQn3K6m",
      sequenceNo:60807
    },
    transactionId:"9RWohCo7My4GGkxRETAJ6M",
    blockTimestamp:2019-09-18T17:00:14.601000001Z,
    blockHash:{{6Pk9KDYJd38ci09oaHxx0D2grtgh4QBBqbDS6i9quX8=}},
    entriesHash:{{r5YoH6+NXDXxgoRzPREGAWJfn73KlZE0eTfbTxZWUDU=}},
    previousBlockHash:{{K3ti0Agk7DEponywKcQCPRYVHb5RuyxdmQFTfrloptA=}},
    entriesHashList:[
      {{pbzvz6ofJC7mD2jvgfyrY/VtR01zIZHoWy8T1VcxlGo=}},
      {{k2brC23DLMercmiOWHiURaGwHu0mQtLzdNPuviE2rcs=}},
      {{hvw1EV8k4oOkIO36kblO/+UUSFUQqCanKuDGr0aP9nQ=}},
      {{ZrLbkyzDcpJ9KWsZMZqRuKUkG/czLIJ4US+K5E31b+Q=}}
    ],
    transactionInfo:{
      statements:[
        {
          statement:"SELECT * FROM Person WHERE GovId = ?",
          startTime:2019-09-18T17:00:14.587Z,
          statementDigest:{{p4Dn0DiuYD3Xm9UQQ75YLwmoMbSfJmopOmTfMnXs26M=}}
        },
        {
          statement:"INSERT INTO Person ?",
          startTime:2019-09-18T17:00:14.594Z,
          statementDigest:{{klMLkLfa5VJqk6JUPtHkQpOsDdG4HmuUaq/VaApQflU=}}
        },
        {
          statement:"INSERT INTO VehicleRegistration ?",
          startTime:2019-09-18T17:00:14.598Z,
          statementDigest:{{B0gO9BWVNrzRYFoe7t+GVLpJ6uZcLKf5t/chkfRhspI=}}
        }
      ],
      documents:{
        '7z2OpEBgVCvCtwvx4a2JGn':{
          tableName:"Person",
          tableId:"LSkFkQvkIOjCmpTZpkfpn9",
          statements:[1]
        },
        'K0FpsSLpydLDr7hi6KUzqk':{
          tableName:"VehicleRegistration",
          tableId:"Ad3A07z0ZffC7Gpso7BXyO",
          statements:[2]
        }
```

```
      }
    },
    revisionSummaries:[
      {
        hash:{{uDthuiqSy4FwjZssyCiyFd90XoPSlIwomHBdF/OrmkE=}},
        documentId:"7z2OpEBgVCvCtwvx4a2JGn"
      },
      {
        hash:{{qJID/amu0gN3dpG5Tg0FfIFTh/U5yFkfT+g/O6k5sPM=}},
        documentId:"K0FpsSLpydLDr7hi6KUzqk"
      }
    ]
  }
}
```

Im `revisionSummaries`-Feld verfügen einige Revisionen möglicherweise über keine `documentId`. Dabei handelt es sich um interne Systemrevisionen, die keine Benutzerdaten enthalten. Ein QLDB-Stream enthält diese Revisionen in den jeweiligen Blockübersichtsdatensätzen, da die Hashes dieser Revisionen Teil der vollständigen Hash-Kette des Journals sind. Die vollständige Hashkette ist für die kryptografische Verifizierung erforderlich.

Nur die Revisionen, die über eine Dokument-ID verfügen, werden, wie im folgenden Abschnitt beschrieben, in separaten Revisionsdetaildatensätzen veröffentlicht.

Revisionsdetaildatensätze

Ein Revisionsdetaildatensatz stellt eine Dokumentversion dar, die für Ihr Journal festgeschrieben ist. Die Nutzlast enthält alle Attribute aus der festgeschriebenen Ansicht der Revision, zusammen mit dem zugeordneten Tabellennamen und der Tabellen-ID. Im Folgenden finden Sie ein Beispiel für einen Revisionsdatensatz mit Beispieldaten.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"REVISION_DETAILS",
  payload:{
    tableInfo:{
      tableName:"VehicleRegistration",
      tableId:"Ad3A07z0ZffC7Gpso7BXyO"
    },
    revision:{
```

```
      blockAddress:{
        strandId:"ElYL30RGoqrFCbbaQn3K6m",
        sequenceNo:60807
      },
      hash:{{qJID/amu0gN3dpG5Tg0FfIFTh/U5yFkfT+g/O6k5sPM=}},
      data:{
        VIN:"1N4AL11D75C109151",
        LicensePlateNumber:"LEWISR261LL",
        State:"WA",
        City:"Seattle",
        PendingPenaltyTicketAmount:90.25,
        ValidFromDate:2017-08-21,
        ValidToDate:2020-05-11,
        Owners:{
          PrimaryOwner:{PersonId:"7z2OpEBgVCvCtwvx4a2JGn"},
          SecondaryOwners:[]
        }
      },
      metadata:{
        id:"K0FpsSLpydLDr7hi6KUzqk",
        version:0,
        txTime:2019-09-18T17:00:14.602Z,
        txId:"9RWohCo7My4GGkxRETAJ6M"
      }
    }
  }
}
```

Umgang mit Duplikaten und out-of-order Datensätzen

QLDB-Streams können Duplikate und out-of-order Datensätze in Kinesis Data Streams veröffentlichen. Daher muss eine Verbraucheranwendung möglicherweise eine eigene Logik implementieren, um solche Szenarien zu identifizieren und zu behandeln. Die Datensätze für Blockzusammenfassung und Revisionsdetails enthalten Felder, die Sie für diesen Zweck verwenden können. In Kombination mit den Funktionen nachgeschalteter Dienste können diese Felder sowohl eine eindeutige Identität als auch eine strenge Reihenfolge für die Datensätze angeben.

Betrachten Sie beispielsweise einen Stream, der QLDB in einen OpenSearch Index integriert, um Volltextsuchfunktionen für Dokumente bereitzustellen. In diesem Anwendungsfall müssen Sie die Indizierung veralteter (out-of-order) Revisionen eines Dokuments vermeiden. Um die Reihenfolge und Deduplizierung zu erzwingen, können Sie die Dokument-ID und die externen

OpenSearchVersionsfelder in zusammen mit den Dokument-ID- und Versionsfeldern in einem Datensatz mit Revisionsdetails verwenden.

Ein Beispiel für die Deduplizierungslogik in einer Beispielanwendung, die QLDB in Amazon OpenSearch Service integriert, finden Sie im GitHub Repository aws-samples/amazon-qldb-streaming-amazon-opensearch-service-sample-python.

Stream-Berechtigungen in QLDB

Bevor Sie einen Amazon-QLDB-Stream erstellen, müssen Sie QLDB Schreibberechtigungen für die angegebene Amazon Kinesis-Data-Streams-Ressource erteilen. Wenn Sie einen Kunden verwenden, der für die serverseitige Verschlüsselung Ihres Kinesis-Streams verwaltet wird AWS KMS key , müssen Sie QLDB auch die Berechtigungen zur Verwendung Ihres angegebenen symmetrischen Verschlüsselungsschlüssels erteilen. Kinesis Data Streams unterstützt keine asymmetrischen KMS-Schlüssel.

Um Ihrem QLDB-Stream die erforderlichen Berechtigungen bereitzustellen, können Sie QLDB eine IAM-Servicerolle mit den entsprechenden Berechtigungsrichtlinien übernehmen lassen. Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

> **Note**
>
> Um beim Anfordern eines Journalstreams eine Rolle an QLDB zu übergeben, müssen Sie über Berechtigungen zum Ausführen der `iam:PassRole`-Aktion für die IAM-Rollenressource verfügen. Dies gilt zusätzlich zur -`qldb:StreamJournalToKinesis`Berechtigung für die QLDB-Stream-Subressource.
>
> Informationen zum Steuern des Zugriffs auf QLDB mit IAM finden Sie unter Funktionsweise von Amazon QLDB mit IAM. Ein Beispiel für eine QLDB-Richtlinie finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

In diesem Beispiel erstellen Sie eine Rolle, die es QLDB ermöglicht, Datensätze in Ihrem Namen in einen Kinesis-Datenstrom zu schreiben. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

Wenn Sie zum AWS-Konto ersten Mal ein QLDB-Journal in Ihrem streamen, müssen Sie zunächst eine IAM-Rolle mit den entsprechenden Richtlinien erstellen, indem Sie wie folgt vorgehen. Oder Sie können die QLDB-Konsole verwenden, um die Rolle automatisch für Sie zu erstellen. Andernfalls können Sie eine Rolle auswählen, die Sie zuvor erstellt haben.

Themen

Erstellen einer Berechtigungsrichtlinie

Führen Sie die folgenden Schritte aus, um eine Berechtigungsrichtlinie für einen QLDB-Stream zu erstellen. Dieses Beispiel zeigt eine Kinesis-Data-Streams-Richtlinie, die QLDB Berechtigungen zum Schreiben von Datensätzen in den angegebenen Kinesis-Datenstrom gewährt. Falls zutreffend, zeigt das Beispiel auch eine Schlüsselrichtlinie, die es QLDB ermöglicht, Ihren KMS-Schlüssel mit symmetrischer Verschlüsselung zu verwenden.

Weitere Informationen zu Kinesis-Data-Streams-Richtlinien finden Sie unter Steuern des Zugriffs auf Amazon Kinesis-Data-Streams-Ressourcen mit IAM und Berechtigungen zur Verwendung von benutzergenerierten KMS-Schlüsseln im Entwicklerhandbuch für Amazon Kinesis Data Streams. Weitere Informationen zu AWS KMS Schlüsselrichtlinien finden Sie unter Verwenden von Schlüsselrichtlinien in AWS KMS im AWS Key Management Service -Entwicklerhandbuch.

> **Note**
>
> Ihr Kinesis-Datenstrom und Ihr KMS-Schlüssel müssen sich beide im selben AWS-Region und Konto wie Ihr QLDB-Ledger befinden.

So verwenden Sie den JSON-Richtlinieneditor zum Erstellen einer Richtlinie

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste auf der linken Seite auf Policies (Richtlinien).

   Wenn Sie zum ersten Mal Policies (Richtlinien) auswählen, erscheint die Seite Welcome to Managed Policies (Willkommen bei verwalteten Richtlinien). Wählen Sie Get Started.
3. Wählen Sie oben auf der Seite Create policy (Richtlinie erstellen) aus.

4. Wählen Sie den Tab JSON.
5. Geben Sie ein JSON-Richtliniendokument ein.
   - Wenn Sie einen vom Kunden verwalteten KMS-Schlüssel für die serverseitige Verschlüsselung Ihres Kinesis-Streams verwenden, verwenden Sie das folgende Beispielrichtliniendokument. Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012kinesis-stream-name*, und *1234abcd-12ab-34cd-56ef-1234567890ab* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBStreamKinesisPermissions",
            "Action": [ "kinesis:PutRecord*", "kinesis:DescribeStream",
 "kinesis:ListShards" ],
            "Effect": "Allow",
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/kinesis-
stream-name"
        },
        {
            "Sid": "QLDBStreamKMSPermission",
            "Action": [ "kms:GenerateDataKey" ],
            "Effect": "Allow",
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

   - Verwenden Sie andernfalls das folgende Beispielrichtliniendokument. Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012* und *kinesis-stream-name* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBStreamKinesisPermissions",
            "Action": [ "kinesis:PutRecord*", "kinesis:DescribeStream",
 "kinesis:ListShards" ],
```

```
            "Effect": "Allow",
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/kinesis-
stream-name"
        }
    ]
}
```

6. Wählen Sie Richtlinie prüfen.

> **Note**
>
> Sie können jederzeit zwischen den Registerkarten Visual editor (Visueller Editor) und JSON wechseln. Wenn Sie jedoch Änderungen vornehmen oder auf der Registerkarte Visueller Editor die Option Richtlinie überprüfen auswählen, strukturiert IAM möglicherweise Ihre Richtlinie neu, um sie für den visuellen Editor zu optimieren. Weitere Informationen finden Sie unter Richtlinienrestrukturierung im IAM-Benutzerhandbuch.

7. Geben Sie auf der Seite Review policy (Richtlinie überprüfen) unter Name einen Namen und unter Description (Beschreibung) eine optionale Beschreibung für die Richtlinie ein, die Sie erstellen. Überprüfen Sie unter Summary die Richtlinienzusammenfassung, um die Berechtigungen einzusehen, die von Ihrer Richtlinie gewährt werden. Wählen Sie dann Create policy aus, um Ihre Eingaben zu speichern.

Erstellen einer IAM-Rolle

Nachdem Sie eine Berechtigungsrichtlinie für Ihren QLDB-Stream erstellt haben, können Sie eine IAM-Rolle erstellen und ihr Ihre Richtlinie anfügen.

So erstellen Sie die Servicerolle für QLDB (IAM-Konsole)

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Klicken Sie im Navigationsbereich der IAM-Konsole auf Rollen, und wählen Sie dann Rolle erstellen.
3. Wählen Sie für Vertrauenswürdige Entität die Option AWS-Service aus.
4. Wählen Sie für Service oder Anwendungsfall QLDB und dann den QLDB-Anwendungsfall aus.
5. Wählen Sie Weiter aus.

6. Aktivieren Sie das Kontrollkästchen neben der Richtlinie, die Sie in den vorherigen Schritten erstellt haben.
7. (Optional) Legen Sie eine Berechtigungsgrenze fest. Dies ist ein erweitertes Feature, das für Servicerollen verfügbar ist, aber nicht für servicegebundene Rollen.

   a. Öffnen Sie den Abschnitt Berechtigungsgrenze festlegen und wählen Sie dann Berechtigungsgrenze verwenden aus, um die maximalen Rollenberechtigungen zu steuern.

      IAM enthält eine Liste der AWS von verwalteten und vom Kunden verwalteten Richtlinien in Ihrem Konto.

   b. Wählen Sie die Richtlinie aus, die für eine Berechtigungsgrenze verwendet werden soll.
8. Wählen Sie Weiter aus.
9. Geben Sie einen Rollennamen oder ein Suffix für den Rollennamen ein, um den Zweck der Rolle zu identifizieren.

   > **Important**
   >
   > Beachten Sie beim Benennen einer Rolle Folgendes:
   >
   > - Rollennamen müssen innerhalb Ihres eindeutig sein AWS-Kontound können nicht durch Groß- und Kleinschreibung eindeutig gemacht werden.
   >
   >   Erstellen Sie beispielsweise keine Rollen mit den Namen **`PRODROLE`** und **`prodrole`**. Wenn ein Rollenname in einer Richtlinie oder als Teil eines ARN verwendet wird, wird die Groß- und Kleinschreibung beachtet. Wenn Kunden in der Konsole jedoch ein Rollenname angezeigt wird, z. B. während des Anmeldevorgangs, wird die Groß- und Kleinschreibung des Rollennamens nicht beachtet.
   > - Sie können den Namen der Rolle nach ihrer Erstellung nicht mehr bearbeiten, da andere Entitäten möglicherweise auf die Rolle verweisen.
10. (Optional) Geben Sie unter Beschreibung eine Beschreibung für die Rolle ein.
11. (Optional) Um die Anwendungsfälle und Berechtigungen für die Rolle zu bearbeiten, wählen Sie in den Abschnitten Schritt 1: Vertrauenswürdige Entitäten auswählen oder Schritt 2: Berechtigungen hinzufügen die Option Bearbeiten aus.
12. (Optional) Um die Rolle zu identifizieren, zu organisieren oder zu suchen, fügen Sie Tags als Schlüssel-Wert-Paare hinzu. Weitere Informationen zur Verwendung von Tags in IAM finden Sie unter Markieren von IAM-Ressourcen im IAM-Benutzerhandbuch.

13. Prüfen Sie die Rolle und klicken Sie dann auf Create Role (Rolle erstellen).

Das folgende JSON-Dokument ist ein Beispiel für eine Vertrauensrichtlinie, die es QLDB ermöglicht, eine IAM-Rolle mit bestimmten Berechtigungen anzunehmen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "qldb.amazonaws.com"
            },
            "Action": [ "sts:AssumeRole" ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:qldb:us-
east-1:123456789012:stream/myExampleLedger/*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

> **Note**
>
> Dieses Beispiel für eine Vertrauensrichtlinie zeigt, wie Sie die `aws:SourceAccount` globalen Bedingungskontextschlüssel `aws:SourceArn` und verwenden können, um das Confused-Deputy-Problem zu vermeiden. Mit dieser Vertrauensrichtlinie kann QLDB die Rolle für jeden QLDB-Stream im Konto `myExampleLedger` nur `123456789012` für das Ledger übernehmen.
>
> Weitere Informationen finden Sie unter Serviceübergreifende Confused-Deputy-Prävention.

Nachdem Sie Ihre IAM-Rolle erstellt haben, kehren Sie zur QLDB-Konsole zurück und aktualisieren Sie die Seite QLDB-Stream erstellen, damit sie Ihre neue Rolle finden kann.

Häufige Fehler für Journalstreams in QLDB

In diesem Abschnitt werden Laufzeitfehler beschrieben, die von Amazon QLDB für Journalstream-Anforderungen ausgelöst werden.

Im Folgenden finden Sie eine Liste mit allgemeinen Ausnahmen, die vom Service zurückgegeben werden. Jede Ausnahme enthält die spezifische Fehlermeldung, gefolgt von einer kurzen Beschreibung und Vorschlägen für mögliche Lösungen.

AccessDeniedException

Meldung: Benutzer: *userARN* ist nicht berechtigt, Folgendes auszuführen: iam:PassRole on-Ressource: *roleARN*

Sie sind nicht berechtigt, eine IAM-Rolle an den QLDB-Service zu übergeben. QLDB benötigt eine Rolle für alle Journalstream-Anforderungen, und Sie müssen über Berechtigungen verfügen, um diese Rolle an QLDB zu übergeben. Die Rolle stellt QLDB Schreibberechtigungen in der von Ihnen angegebenen Amazon Kinesis Data Streams-Ressource bereit.

Stellen Sie sicher, dass Sie eine IAM-Richtlinie definieren, die die Berechtigung zum Ausführen der `PassRole` API-Operation für die von Ihnen angegebene IAM-Rollenressource für den QLDB-Service (`qldb.amazonaws.com`) gewährt. Eine Beispielrichtlinie finden Sie in Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

IllegalArgumentException

Meldung: QLDB hat einen Fehler beim Validieren von Kinesis Data Streams festgestellt: Antwort von Kinesis: *errorCode errorMessage*

Eine mögliche Ursache für diesen Fehler ist, dass die bereitgestellte Kinesis Data Streams-Ressource nicht existiert. Oder QLDB verfügt nicht über genügend Berechtigungen, um Datensätze in den angegebenen Kinesis-Datenstrom zu schreiben.

Stellen Sie sicher, dass der Kinesis-Datenstrom, den Sie in Ihrer Stream-Anforderung angeben, korrekt ist. Weitere Informationen finden Sie unter Erstellen und Aktualisieren von Datenströmen im Entwicklerhandbuch für Amazon Kinesis Data Streams.

Stellen Sie außerdem sicher, dass Sie eine Richtlinie für den angegebenen Kinesis-Datenstrom definieren, die dem QLDB-Service (`qldb.amazonaws.com`) Berechtigungen für die folgenden Aktionen gewährt. Weitere Informationen finden Sie unter Stream-Berechtigungen.

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

Meldung: Unerwartete Antwort von Kinesis Data Streams bei der Validierung der Kinesis-Konfiguration. Antwort von Kinesis: *`errorCode errorMessage`*

Der Versuch, Datensätze in den bereitgestellten Kinesis-Datenstrom zu schreiben, ist mit der bereitgestellten Kinesis-Fehlerantwort fehlgeschlagen. Weitere Informationen zu möglichen Ursachen finden Sie unter Fehlerbehebung bei Produzenten von Amazon Kinesis Data Streams im Entwicklerhandbuch für Amazon Kinesis Data Streams.

IllegalArgumentException

Meldung: Start date must not be greater than end date.

Sowohl `InclusiveStartTime` als auch `ExclusiveEndTime` müssen im Datums- und Zeitformat ISO 8601 und in koordinierter Weltzeit (Coordinated Universal Time, UTC) angegeben werden.

IllegalArgumentException

Meldung: Startdatum kann nicht in der Zukunft liegen.

Sowohl `InclusiveStartTime` als auch `ExclusiveEndTime` müssen im Datums- und Zeitformat `ISO 8601` und in koordinierter Weltzeit (Coordinated Universal Time, UTC) angegeben werden.

LimitExceededException

Meldung: Grenzwert von 5 parallel ausgeführten Journal-Streams zu Kinesis Data Streams überschritten

QLDB erzwingt ein Standardlimit von fünf gleichzeitigen Journalstreams.

Buchverwaltung in Amazon QLDB

In diesem Kapitel wird beschrieben, wie Sie die QLDB-APIAWS Command Line Interface (AWS CLI) verwenden undAWS CloudFormation Ledger-Verwaltungsvorgänge in Amazon QLDB durchführen.

Sie können auch die entsprechenden Aufgaben mithilfe der AWS Management Console durchführen. Weitere Informationen finden Sie unter Zugreifen auf Amazon QLDB über die Konsole.

Themen

- Grundfunktionen für Amazon QLDB-Ledgers
- Erstellen von Amazon QLDB-Ressourcen mitAWS CloudFormation
- Markieren von Amazon-QLDB Ressourcen

Grundfunktionen für Amazon QLDB-Ledgers

Sie können die QLDB-API oder dieAWS Command Line Interface (AWS CLI) verwenden, um Ledger in Amazon QLDB zu erstellen, zu aktualisieren und zu löschen. Sie können auch alle Ledger in Ihrem Konto auflisten oder Informationen zu einem bestimmten Ledger abrufen.

Die folgenden Themen enthalten Kurzcodebeispiele, die allgemeine Schritte für Ledger-Operationen unter Verwendung vonAWS SDK for Java undAWS CLI

Themen

- Ein Ledger erstellen
- Beschreiben eines Ledgers
- Ein Hauptbuch aktualisieren
- Einen Ledger-Berechtigungsmodus aktualisieren
- Löschen eines Ledgers
- Ledger auflisten

Codebeispiele, die diese Operationen in einer vollständigen Beispielanwendung demonstrieren, finden Sie in den folgendenErste Schritte mit dem Treiber Tutorials und GitHub Repositorys:

- Java: Anleitung | GitHub Repositorium

- Node.js: Anleitung | GitHub Repositorium
- Python: Anleitung | GitHub Repositorium

Ein Ledger erstellen

Verwenden Sie die`CreateLedger` Operation, um ein Hauptbuch in Ihrem zu erstellenAWS-Konto. Sie müssen die folgenden Informationen angeben:

- Ledger-Name — Der Name des Ledgers, das Sie in Ihrem Konto erstellen möchten. Der Name muss unter allen Ihren Ledgern im aktuellen eindeutig seinAWS-Region.

  Benennungseinschränkungen für Ledger-Namen sind in Kontingente und Limits in Amazon QLDB definiert.

- Berechtigungsmodus — Der Berechtigungsmodus, der dem Ledger zugewiesen werden soll. Wählen Sie eine der folgenden Optionen:

  - Allow all — Ein Legacy-Berechtigungsmodus, der die Zugriffskontrolle mit Granularität auf API-Ebene für Ledgers ermöglicht.

    Dieser Modus ermöglicht Benutzern, die `SendCommand`-API-Berechtigung für dieses Ledger zum Ausführen aller PartiQL-Befehle (daher `ALLOW_ALL`) auf beliebigen Tabellen im angegebenen Ledger auszuführen. Dieser Modus ignoriert alle IAM-Berechtigungs-Richtlinien auf Tabellen- oder Befehls-Ebene, die Sie für das Ledger erstellen.

  - Standard — (Empfohlen) Ein Berechtigungsmodus, der Zugriffskontrolle mit feinerer Granularität für Ledger, Tabellen und PartiQL-Befehle ermöglicht. Wir empfehlen dringend, diesen Berechtigungsmodus, um die Sicherheit Ihrer Ledger-Daten zu maximieren.

    Standardmäßig verweigert dieser Modus alle Anforderungen zur Ausführung von PartiQL-Befehlen für Tabellen in diesem Ledger. Um PartiQL-Befehle zuzulassen, müssen Sie IAM-Berechtigungs-Richtlinien für bestimmte Tabellen-Ressourcen und PartiQL-Aktionen erstellen, zusätzlich zu den`SendCommand` API-Berechtigungs-Richtlinien für den Ledger. Weitere Informationen finden Sie unter Erste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

- Löschschutz — (Optional) Das Flag, das verhindert, dass ein Ledger von einem beliebigen Benutzer gelöscht wird. Wenn Sie es während der Ledger-Erstellung nicht angeben, ist diese Funktion standardmäßig aktiviert (`true`).

Wenn der Löschschutz aktiviert ist, müssen Sie ihn zuerst deaktivieren, bevor Sie das Ledger löschen können. Sie können sie deaktivieren, indem Sie mit der Operation `UpdateLedger` das Flag auf `false` einstellen.

- AWS KMS key— (Optional) Der Schlüssel inAWS Key Management Service (AWS KMS), der für die Verschlüsselung ruhender Daten verwendet werden soll. Wählen Sie einen der folgenden Typen vonAWS KMS keys:
  - AWSeigener KMS-Schlüssel — Einen KMS-Schlüssel verwenden, der gehört und inAWS Ihrem Namen verwaltet wird.

    Wenn Sie diesen Parameter während der Ledger-Erstellung nicht definieren, verwendet das Ledger standardmäßig diesen Schlüsseltyp. Sie können die Zeichenfolge auch verwenden`AWS_OWNED_KMS_KEY`, um diesen Schlüsseltyp anzugeben. Diese Option erfordert keine zusätzliche Einrichtung.
  - Kundenverwalteter KMS-Schlüssel — Einen symmetrischen KMS-Verschlüsselungsschlüssel in Ihrem Konto verwenden, den Sie erstellen, besitzen und verwalten. QLDB unterstützt keine asymmetrischen Schlüssel.

    Für diese Option müssen Sie einen KMS-Schlüssel erstellen oder einen vorhandenen Schlüssel in Ihrem Konto verwenden. Anweisungen zum Erstellen eines vom Kunden verwalteten Schlüssels finden Sie unter Erstellen von KMS-Schlüsseln mit symmetrischer Verschlüsselung im AWS Key Management ServiceEntwicklerhandbuch.

    Sie können einen vom Kunden verwalteten KMS-Schlüssel angeben, indem Sie eine ID, einen Alias oder einen Amazon-Ressourcennamen (ARN) verwenden. Weitere Informationen finden Sie unter Key Identifiers (KeyId) im AWS Key Management ServiceDeveloper Guide.

    > **Note**
    >
    > Regionsübergreifende Schlüssel werden nicht unterstützt. Der angegebene KMS-Schlüssel muss sich im gleichenAWS-Region wie Ihr Ledger befinden.

  Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand in Amazon QLDB.
- Tags — (Optional) Fügen Sie dem Ledger Metadaten hinzu, indem Sie Tags als Schlüssel-Wert-Paare anfügen. Sie können Ihrem Ledger Tags hinzufügen, um sie zu organisieren und zu identifizieren. Weitere Informationen finden Sie unter Markieren von Amazon-QLDB Ressourcen.

Das Ledger kann erst verwendet werden, wenn QLDB es erstellt und seinen Status auf gesetzt hat`ACTIVE`.

Erstellen eines Ledgers (Java)

So erstellen Sie einen Ledger mit dem AWS SDK for Java:

1. Erstellen Sie eine Instance der `AmazonQLDB`-Klasse.
2. Erstellen Sie eine Instance der `CreateLedgerRequest`-Klasse, um die Anforderungsinformationen bereitzustellen.

   Sie müssen den Ledger-Namen und einen Berechtigungsmodus angeben.
3. Führen Sie die `createLedger`-Methode aus, indem das Anforderungsobjekt als Parameter festgelegt wird.

Die `createLedger`-Anforderung gibt ein `CreateLedgerResult`-Objekt zurück, das Informationen über den Ledger enthält. Im nächsten Abschnitt finden Sie ein Beispiel für die Verwendung der Operation `DescribeLedger`, um den Status Ihres Ledgers nach dem Erstellen zu überprüfen.

Die folgenden Beispiele verdeutlichen die vorherigen Schritte.

Example — Standardkonfigurationseinstellungen verwenden

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
CreateLedgerRequest request = new CreateLedgerRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
CreateLedgerResult result = client.createLedger(request);
```

> **Note**
>
> Das Ledger verwendet die folgenden Standardeinstellungen, wenn Sie sie nicht angeben:
>
> - Löschschutz — Aktiviert (`true`).
> - KMS-Schlüssel —AWS eigener KMS-Schlüssel.

Example — Deaktivieren Sie den Löschschutz, verwenden Sie einen vom Kunden verwalteten KMS-Schlüssel und hängen Sie Tags an

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();

Map<String, String> tags = new HashMap<>();
tags.put("IsTest", "true");
tags.put("Domain", "Test");

CreateLedgerRequest request = new CreateLedgerRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD)
        .withDeletionProtection(false)
        .withKmsKey("arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab")
        .withTags(tags);
CreateLedgerResult result = client.createLedger(request);
```

Ein Ledger erstellen (AWS CLI)

Erstellen Sie ein neues Ledger`vehicle-registration` mit dem Namen unter Verwendung der Standardkonfigurationseinstellungen.

Example

```
aws qldb create-ledger --name vehicle-registration --permissions-mode STANDARD
```

> **Note**
>
> Das Ledger verwendet die folgenden Standardeinstellungen, wenn Sie sie nicht angeben:
>
> - Löschschutz — Aktiviert (`true`).
> - KMS-Schlüssel —AWS eigener KMS-Schlüssel.

Oder erstellen Sie ein neues Ledger`vehicle-registration` mit deaktiviertem Löschschutz, mit einem bestimmten, vom Kunden verwalteten KMS-Schlüssel und mit bestimmten Tags.

Example

```
aws qldb create-ledger \
```

```
    --name vehicle-registration \
    --no-deletion-protection \
    --permissions-mode STANDARD \
    --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --tags IsTest=true,Domain=Test
```

Ein Ledger erstellen (AWS CloudFormation)

Sie können auch eine AWS CloudFormationVorlage verwenden, um Ledger zu erstellen. Weitere Informationen finden Sie in der AWS::QLDB::LedgerRessource im AWS CloudFormationBenutzerhandbuch.

Beschreiben eines Ledgers

Um Details über einen Ledger anzuzeigen, verwenden Sie die Operation `DescribeLedger`. Sie müssen den Ledger-Namen angeben. Die Ausgabe von `DescribeLedger` erfolgt in demselben Format wie bei `CreateLedger`. Dazu gehören folgende Informationen:

- Ledger-Name — Der Name des Ledgers, das Sie beschreiben möchten.
- ARN — Der Amazon-Ressourcenname (ARN) für das Ledger im folgenden Format.

  ```
  arn:aws:qldb:aws-region:account-id:ledger/ledger-name
  ```

- Löschschutz — Die Flagge, die angibt, ob die Löschschutzfunktion aktiviert ist.
- Erstellungsdatum und -zeit — Datum und Uhrzeit im Epochenformat der Erstellung des Ledgers.
- Status — Der aktuelle Status des Ledgers. Dabei kann es sich um einen der folgenden Werte handeln:
  - `CREATING`
  - `ACTIVE`
  - `DELETING`
  - `DELETED`
- Berechtigungsmodus — Der Berechtigungsmodus, der dem Ledger zugewiesen ist. Dabei kann es sich um einen der folgenden Werte handeln:
  - `ALLOW_ALL`— Ein Legacy-Berechtigungsmodus, der die Zugriffskontrolle mit Granularität auf API-Ebene für Ledgers ermöglicht.

  - `STANDARD`— Ein Berechtigungsmodus, der Zugriffskontrolle mit feinerer Granularität für Ledger, Tabellen und PartiQL-Befehle ermöglicht.
- Verschlüsselungsbeschreibung — Informationen zur Verschlüsselung von Daten im Ruhezustand im Ledger. Dies umfasst die folgenden Elemente:
  - AWS KMS keyARN — Der ARN des vom Kunden verwalteten KMS-Schlüssels, den das Ledger für die Verschlüsselung im Ruhezustand verwendet. Wenn dieser nicht definiert ist, verwendet das Ledger einenAWS eigenen KMS-Schlüssel zur Verschlüsselung.
  - Verschlüsselungsstatus — Der aktuelle Status der Verschlüsselung im Ruhezustand für das Ledger. Dabei kann es sich um einen der folgenden Werte handeln:
    - `ENABLED`— Die Verschlüsselung ist mit dem angegebenen Schlüssel vollständig aktiviert.
    - `UPDATING`— Die angegebene Schlüsseländerung wird aktiv verarbeitet.

      Die wichtigsten Änderungen in QLDB sind asynchron. Während der Bearbeitung der wichtigsten Änderung ist das Hauptbuch vollständig zugänglich, ohne dass sich dies auf die Leistung auswirkt. Die Zeit, die für die Aktualisierung eines Schlüssels benötigt wird, hängt von der Ledger-Größe ab.
    - `KMS_KEY_INACCESSIBLE`— Auf den angegebenen, vom Kunden verwalteten KMS-Schlüssel kann nicht zugegriffen werden, und das Hauptbuch ist beeinträchtigt. Entweder wurde der Schlüssel deaktiviert oder gelöscht, oder die Zuschüsse für den Schlüssel wurden widerrufen. Wenn ein Hauptbuch beeinträchtigt ist, ist es nicht zugänglich und akzeptiert keine Lese- oder Schreibanfragen.

      Ein beeinträchtigtes Ledger kehrt automatisch in den aktiven Zustand zurück, nachdem Sie die Zuschüsse für den Schlüssel wiederhergestellt haben oder nachdem Sie den deaktivierten Schlüssel erneut aktiviert haben. Das Löschen eines vom Kunden verwalteten KMS-Schlüssels kann jedoch nicht rückgängig gemacht werden. Nach dem Löschen eines Schlüssels können Sie nicht mehr auf die Ledger zugreifen, die mit diesem Schlüssel geschützt sind, und die Daten können nicht dauerhaft wiederhergestellt werden.
  - Nicht zugänglichAWS KMS key — Datum und Uhrzeit im Epochenformat, an dem der KMS-Schlüssel im Fehlerfall zum ersten Mal nicht mehr zugänglich war.

    Dies ist undefiniert, wenn auf den KMS-Schlüssel zugegriffen werden kann.

Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand in Amazon QLDB.

> **Note**
>
> Nachdem Sie ein QLDB-Ledger erstellt haben, ist es einsatzbereit, wenn sich sein Status von`CREATING` bis ändert`ACTIVE`.

Beschreibung eines Ledgers (Java)

So beschreiben Sie einen Ledger mit dem AWS SDK for Java:

1. Erstellen Sie eine Instance der `AmazonQLDB`-Klasse. Oder Sie können dieselbe Instance des `AmazonQLDB`-Clients verwenden, die Sie für die Anforderung `CreateLedger` instanziiert haben.
2. Erstellen Sie eine Instance der `DescribeLedgerRequest`-Klasse und geben Sie den Namen des Ledgers an, den Sie löschen möchten.
3. Führen Sie die `describeLedger`-Methode aus, indem das Anforderungsobjekt als Parameter festgelegt wird.
4. Die Anforderung `describeLedger` gibt ein `DescribeLedgerResult`-Objekt zurück, das aktuelle Informationen über den Ledger enthält.

Im folgenden Codebeispiel werden die vorherigen Schritte veranschaulicht. Sie können die `describeLedger`-Methode des Clients aufrufen, um die Ledger-Informationen jederzeit abzurufen.

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
DescribeLedgerRequest request = new DescribeLedgerRequest().withName(ledgerName);
DescribeLedgerResult result = client.describeLedger(request);
System.out.printf("%s: ARN: %s \t State: %s \t CreationDateTime: %s \t
 DeletionProtection: %s
                  \t PermissionsMode: %s \t EncryptionDescription: %s",
        result.getName(),
        result.getArn(),
        result.getState(),
        result.getCreationDateTime(),
        result.getDeletionProtection(),
        result.getPermissionsMode(),
        result.getEncryptionDescription());
```

Beschreibung eines Ledgers (AWS CLI)

Beschreiben Sie den `vehicle-registration`-Ledger, den Sie gerade erstellt haben.

Example

```
aws qldb describe-ledger --name vehicle-registration
```

Ein Hauptbuch aktualisieren

Mit`UpdateLedger` diesem Vorgang können Sie derzeit die folgenden Konfigurationseinstellungen für ein vorhandenes Ledger ändern:

- Löschschutz — Das Flag, das verhindert, dass ein Ledger von einem beliebigen Benutzer gelöscht wird. Wenn diese Funktion aktiviert ist, müssen Sie sie zuerst deaktivieren, indem Sie die Flagge auf setzen,`false` bevor Sie das Ledger löschen können.

  Wenn Sie diesen Parameter nicht definieren, werden keine Änderungen an den Löschschutzeinstellungen des Ledgers vorgenommen.
- AWS KMS key— Der Schlüssel inAWS Key Management Service (AWS KMS), der für die Verschlüsselung von Daten im Speicher verwendet werden soll. Wenn Sie diesen Parameter nicht definieren, werden keine Änderungen am KMS-Schlüssel des Ledgers vorgenommen.

  > **Note**
  >
  > Amazon QLDB hatAWS KMS keys am 22. Juli 2021 den Support für kundenverwaltete Kunden eingeführt. Alle Ledger, die vor dem Start erstellt wurden, sindAWS-eigene Schlüssel standardmäßig geschützt, können aber derzeit nicht für die Verschlüsselung im Speicher mithilfe von vom Kunden verwalteten Schlüsseln verwendet werden.
  > Sie können die Erstellungszeit Ihres Ledgers auf der QLDB-Konsole einsehen.

  Wählen Sie eine der folgenden Optionen aus:

  - AWSeigener KMS-Schlüssel — Einen KMS-Schlüssel verwenden, der gehört und inAWS Ihrem Namen verwaltet wird. Um diesen Schlüsseltyp zu verwenden, geben Sie die ZeichenfolgeAWS_OWNED_KMS_KEY für diesen Parameter an. Diese Option erfordert keine zusätzliche Einrichtung.

- Kundenverwalteter KMS-Schlüssel — Einen symmetrischen KMS-Verschlüsselungsschlüssel in Ihrem Konto verwenden, den Sie erstellen, besitzen und verwalten. QLDB unterstützt keine asymmetrischen Schlüssel.

  Für diese Option müssen Sie einen KMS-Schlüssel erstellen oder einen vorhandenen Schlüssel in Ihrem Konto verwenden. Anweisungen zum Erstellen eines vom Kunden verwalteten Schlüssels finden Sie unter Erstellen von KMS-Schlüsseln mit symmetrischer Verschlüsselung im AWS Key Management ServiceEntwicklerhandbuch.

  Sie können einen vom Kunden verwalteten KMS-Schlüssel angeben, indem Sie eine ID, einen Alias oder einen Amazon-Ressourcennamen (ARN) verwenden. Weitere Informationen finden Sie unter Key Identifiers (KeyId) im AWS Key Management ServiceDeveloper Guide.

  > **Note**
  >
  > Regionsübergreifende Schlüssel werden nicht unterstützt. Der angegebene KMS-Schlüssel muss sich im gleichenAWS-Region wie Ihr Ledger befinden.

Die wichtigsten Änderungen in QLDB sind asynchron. Während der Bearbeitung der wichtigsten Änderung ist das Hauptbuch vollständig zugänglich, ohne dass sich dies auf die Leistung auswirkt.

Sie können die Schlüssel so oft wie nötig wechseln, aber die Zeit, die für die Aktualisierung eines Schlüssels benötigt wird, hängt von der Buchgröße ab. Sie können den`DescribeLedger` Vorgang verwenden, um die Verschlüsselung im Ruhezustand zu überprüfen.

Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand in Amazon QLDB.

Die Ausgabe von `UpdateLedger` erfolgt in demselben Format wie bei `CreateLedger`.

Ein Ledger aktualisieren (Java)

So aktualisieren Sie einen Ledger mit dem AWS SDK for Java:

1. Erstellen Sie eine Instance der `AmazonQLDB`-Klasse.
2. Erstellen Sie eine Instance der `UpdateLedgerRequest`-Klasse, um die Anforderungsinformationen bereitzustellen.

   Sie müssen den Ledger-Namen zusammen mit einem neuen booleschen Wert für den Löschschutz oder einem neuen Zeichenfolgenwert für den KMS-Schlüssel angeben.

3. Führen Sie die `updateLedger`-Methode aus, indem das Anforderungsobjekt als Parameter festgelegt wird.

Die folgenden Codebeispiele veranschaulichen die vorherigen Schritte. Die Anforderung `updateLedger` gibt ein `UpdateLedgerResult`-Objekt mit aktualisierten Informationen zum Ledger zurück.

Example — Löschschutz deaktivieren

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withDeletionProtection(false);
UpdateLedgerResult result = client.updateLedger(request);
```

Example — Einen vom Kunden verwalteten KMS-Schlüssel verwenden

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withKmsKey("arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab")
UpdateLedgerResult result = client.updateLedger(request);
```

Example — Verwenden Sie einenAWS eigenen KMS-Schlüssel

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withKmsKey("AWS_OWNED_KMS_KEY")
UpdateLedgerResult result = client.updateLedger(request);
```

Ein Ledger aktualisieren (AWS CLI)

Wenn für Ihren `vehicle-registration`-Ledger der Löschschutz aktiviert ist, müssen Sie ihn zunächst deaktivieren, bevor Sie ihn löschen können.

Example

```
aws qldb update-ledger --name vehicle-registration --no-deletion-protection
```

Sie können auch die Einstellungen für die Verschlüsselung im Speicher ändern, sodass ein vom Kunden verwalteter KMS-Schlüssel verwendet wird.

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Sie können auch die Einstellungen für die Verschlüsselung im Ruhezustand ändern, sodass einAWS eigener KMS-Schlüssel verwendet wird.

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key AWS_OWNED_KMS_KEY
```

Einen Ledger-Berechtigungsmodus aktualisieren

Mit`UpdateLedgerPermissionsMode` diesem Vorgang können Sie den Berechtigungsmodus eines vorhandenen Ledgers ändern. Wählen Sie eine der folgenden Optionen:

- Allow all — Ein Legacy-Berechtigungsmodus, der die Zugriffskontrolle mit Granularität auf API-Ebene für Ledgers ermöglicht.

  Dieser Modus ermöglicht Benutzern, die `SendCommand`-API-Berechtigung für dieses Ledger zum Ausführen aller PartiQL-Befehle (daher `ALLOW_ALL`) auf beliebigen Tabellen im angegebenen Ledger auszuführen. Dieser Modus ignoriert alle IAM-Berechtigungs-Richtlinien auf Tabellen- oder Befehls-Ebene, die Sie für das Ledger erstellen.
- Standard — (Empfohlen) Ein Berechtigungsmodus, der Zugriffskontrolle mit feinerer Granularität für Ledger, Tabellen und PartiQL-Befehle ermöglicht. Wir empfehlen dringend, diesen Berechtigungsmodus, um die Sicherheit Ihrer Ledger-Daten zu maximieren.

  Standardmäßig verweigert dieser Modus alle Anforderungen zur Ausführung von PartiQL-Befehlen für Tabellen in diesem Ledger. Um PartiQL-Befehle zuzulassen, müssen Sie IAM-Berechtigungs-Richtlinien für bestimmte Tabellen-Ressourcen und PartiQL-Aktionen erstellen, zusätzlich zu den`SendCommand` API-Berechtigungs-Richtlinien für den Ledger. Weitere Informationen finden Sie unter Erste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

> **Important**
>
> Bevor Sie in den`STANDARD` Berechtigungsmodus wechseln, müssen Sie zunächst alle erforderlichen IAM-Richtlinien und Tabellen-Tags erstellen, um Störungen für Ihre Benutzer zu vermeiden. Um mehr zu erfahren, fahren Sie mit fortMigration in den Standardberechtigungsmodus.

Aktualisierung eines Ledger-Berechtigungsmodus (Java)

Um einen Ledger-Berechtigungsmodus zu aktualisieren, verwenden Sie dieAWS SDK for Java

1. Erstellen Sie eine Instance der `AmazonQLDB`-Klasse.
2. Erstellen Sie eine Instance der `UpdateLedgerPermissionsModeRequest`-Klasse, um die Anforderungsinformationen bereitzustellen.

   Sie müssen den Ledger-Namen zusammen mit einem neuen Zeichenfolgenwert für den Berechtigungsmodus angeben.
3. Führen Sie die `updateLedgerPermissionsMode`-Methode aus, indem das Anforderungsobjekt als Parameter festgelegt wird.

Die folgenden Codebeispiele veranschaulichen die vorherigen Schritte. Die Anforderung `updateLedgerPermissionsMode` gibt ein `UpdateLedgerPermissionsModeResult`-Objekt mit aktualisierten Informationen zum Ledger zurück.

Example — Weisen Sie den Standardberechtigungsmodus zu

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerPermissionsModeRequest request = new UpdateLedgerPermissionsModeRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
UpdateLedgerPermissionsModeResult result = client.updateLedgerPermissionsMode(request);
```

Einen Ledger-Berechtigungsmodus aktualisieren (AWS CLI)

Weisen Sie Ihrem`vehicle-registration` Ledger den`STANDARD` Berechtigungsmodus zu.

Example

```
aws qldb update-ledger-permissions-mode --name vehicle-registration --permissions-mode
 STANDARD
```

Migration in den Standardberechtigungsmodus

Um in den`STANDARD` Berechtigungsmodus zu migrieren, empfehlen wir, Ihre QLDB-Zugriffsmuster zu analysieren und IAM-Richtlinien hinzuzufügen, die Ihren Benutzern die entsprechenden Berechtigungen für den Zugriff auf ihre Ressourcen gewähren.

Bevor Sie in den`STANDARD` Berechtigungsmodus wechseln, müssen Sie zunächst alle erforderlichen IAM-Richtlinien und Tabellen-Tags erstellen. Andernfalls kann ein Wechsel des Berechtigungsmodus Benutzer unterbrechen, bis Sie entweder die richtigen IAM-Richtlinien erstellt haben oder den Berechtigungsmodus wieder auf zurücksetzen`ALLOW_ALL`. Weitere Informationen zum Erstellen dieser Richtlinien finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

Sie können auch eineAWS verwaltete Richtlinie verwenden, um vollen Zugriff auf alle QLDB-Ressourcen zu gewähren. Die`AmazonQLDBFullAccess` und`AmazonQLDBConsoleFullAccess` verwalteten Richtlinien umfassen alle QLDB-Aktionen, einschließlich aller PartiQL-Aktionen. Das Anhängen einer dieser Richtlinien an einen Principal entspricht dem`ALLOW_ALL` Berechtigungsmodus für diesen Principal. Weitere Informationen finden Sie unter AWS Von verwaltete Richtlinien für Amazon QLDB.

Löschen eines Ledgers

Verwenden Sie die Operation `DeleteLedger`, um einen Ledger und seinen gesamten Inhalt zu löschen. Das Löschen eines Ledgers ist ein unwiederbringlicher Vorgang.

Wenn der Löschschutz für Ihr Ledger aktiviert ist, müssen Sie ihn zuerst deaktivieren, bevor Sie das Ledger löschen können.

Wenn Sie eine `DeleteLedger`-Anforderung ausgeben, ändert sich der Status des Ledgers von `ACTIVE` in `DELETING`. Es kann eine Weile dauern, bis der Ledger gelöscht wird, abhängig von der Menge des verwendeten Speichers. Wenn der`DeleteLedger` Vorgang abschließt, ist das Ledger nicht mehr in QLDB vorhanden.

Löschen eines Ledgers (Java)

So löschen Sie einen Ledger mit dem AWS SDK for Java:

1. Erstellen Sie eine Instance der `AmazonQLDB`-Klasse.
2. Erstellen Sie eine Instance der `DeleteLedgerRequest`-Klasse und geben Sie den Namen des Ledgers an, den Sie löschen möchten.
3. Führen Sie die `deleteLedger`-Methode aus, indem das Anforderungsobjekt als Parameter festgelegt wird.

Im folgenden Codebeispiel werden die vorherigen Schritte veranschaulicht.

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
DeleteLedgerRequest request = new DeleteLedgerRequest().withName(ledgerName);
DeleteLedgerResult result = client.deleteLedger(request);
```

Löschen eines Ledgers (AWS CLI)

Löschen Sie Ihren `vehicle-registration`-Ledger.

Example

```
aws qldb delete-ledger --name vehicle-registration
```

Ledger auflisten

Der`ListLedgers` Vorgang gibt zusammenfassende Informationen aller QLDB-Ledger für den aktuellen StatusAWS-Konto und die Region zurück.

Ledger auflisten (Java)

So listen Sie Ledger mit dem AWS SDK for Java in Ihrem Konto auf:

1. Erstellen Sie eine Instance der `AmazonQLDB`-Klasse.
2. Erstellen Sie eine Instance der `ListLedgersRequest`-Klasse.

Wenn Sie`NextToken` in der Antwort eines früheren`ListLedgers` Anrufs einen Wert für erhalten haben, müssen Sie diesen Wert in dieser Anfrage angeben, um die nächste Ergebnisseite zu erhalten.

3. Führen Sie die `listLedgers`-Methode aus, indem das Anforderungsobjekt als Parameter festgelegt wird.
4. Die Anforderung `listLedgers` gibt ein `ListLedgersResult`-Objekt zurück. Dieses Objekt verfügt über eine Liste von `LedgerSummary`-Objekten und ein Paginierungs-Token, das angibt, ob weitere Ergebnisse verfügbar sind:

   - Wenn `NextToken` leer ist, wurde die letzte Ergebnisseite verarbeitet und es gibt keine weiteren Ergebnisse.
   - Wenn `NextToken` nicht leer ist, sind weitere Ergebnisse verfügbar. Um die nächste Ergebnisseite abzurufen, verwenden Sie den Wert von `NextToken` in einem nachfolgenden `ListLedgers`-Aufruf.

Im folgenden Codebeispiel werden die vorherigen Schritte veranschaulicht.

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
List<LedgerSummary> ledgerSummaries = new ArrayList<>();
String nextToken = null;
do {
    ListLedgersRequest request = new ListLedgersRequest().withNextToken(nextToken);
    ListLedgersResult result = client.listLedgers(request);
    ledgerSummaries.addAll(result.getLedgers());
    nextToken = result.getNextToken();
} while (nextToken != null);
```

Hauptbücher auflisten (AWS CLI)

Listet alle Ledger im aktuellenAWS-Konto und in der Region auf.

Example

```
aws qldb list-ledgers
```

Erstellen von Amazon QLDB-Ressourcen mitAWS CloudFormation

Amazon QLDB ist in integriertAWS CloudFormation, ein Service, der Ihnen hilft, IhreAWS Ressourcen zu modellieren und einzurichten, damit Sie weniger Zeit mit der Erstellung und Verwaltung Ihrer Ressourcen und Infrastruktur verbringen können. Sie erstellen eine Vorlage, in der alle gewünschtenAWS -Ressourcen beschrieben werden (wie QLDB-Ledger) und übernimmtAWS CloudFormation die Bereitstellung und Konfiguration dieser Ressourcen für Sie.

Wenn Sie verwendenAWS CloudFormation, können Sie Ihre Vorlage wiederverwenden, um Ihre QLDB-Ressourcen einheitlich und wiederholt einzurichten. Sie beschreiben Ihre Ressourcen dann einmal und können die gleichen Ressourcen dann in mehreren AWS-Konten-Konten und -Regionen immer wieder bereitstellen.

QLDB undAWS CloudFormation Vorlagen

Um Ressourcen für QLDB und zugehörige Dienste bereitzustellen und zu konfigurieren, müssen Sie AWS CloudFormation-Vorlagen kennen und verstehen. Vorlagen sind formatierte Textdateien in JSON oder YAML. Diese Vorlagen beschreiben die Ressourcen, die Sie in Ihren AWS CloudFormation-Stacks bereitstellen möchten. Wenn Sie noch keine Erfahrungen mit JSON oder YAML haben, können Sie AWS CloudFormation Designer verwenden, der den Einstieg in die Arbeit mit AWS CloudFormation-Vorlagen erleichtert. Weitere Informationen finden Sie unter Was ist AWS CloudFormation-Designer? im AWS CloudFormation-Benutzerhandbuch.

QLDB unterstützt das Erstellen von Ledgern und Journal-Streams inAWS CloudFormation. Weitere Informationen, einschließlich Beispiele für JSON- und YAML-Vorlagen für Ledger und Streams, finden Sie in den folgenden Ressourcentypreferenzen im AWS CloudFormation-Benutzerhandbuch:

- AWS: :QLDB: :Ledger-Referenz
- AWS: :QLDB: :Stream Referenz

Weitere Informationen zu AWS CloudFormation

Weitere Informationen zu AWS CloudFormation finden Sie in den folgenden Ressourcen.

- AWS CloudFormation
- AWS CloudFormation-Benutzerhandbuch
- AWS CloudFormation API Referenz

- AWS CloudFormation-Benutzerhandbuch für die Befehlszeilenschnittstelle

Markieren von Amazon-QLDB Ressourcen

Ein Tag ist eine benutzerdefinierte Attributbezeichnung, die Sie oder AWS einer AWS-Ressource zuweisen. Jedes Tag besteht aus zwei Teilen:

- einem Tag-Schlüssel (z. B. `CostCenter`, `Environment` oder `Project`). Bei Tag-Schlüsseln wird zwischen Groß- und Kleinschreibung unterschieden.
- einem optionalen Feld, dem sogenannten Tag-Wert (z. B. `111122223333` oder `Production`). Ein nicht angegebener Tag-Wert entspricht einer leeren Zeichenfolge. Wie bei Tag-Schlüsseln wird auch bei Tag-Werten zwischen Groß- und Kleinschreibung unterschieden.

Tags sind für folgende Aktivitäten nützlich:

- Identifizieren und Organisieren Ihrer AWS-Ressourcen. Viele AWS-Services unterstützen das Markieren mit Tags (kurz: Tagging). So können Ressourcen aus verschiedenen Services das gleiche Tag zuweisen, um anzugeben, dass die Ressourcen zusammengehören. Sie können beispielsweise das gleiche Tag einem Amazon QLDB Ledger zuweisen, das Sie einem Amazon S3 S3-Bucket zuweisen.
- Überwachen von AWS-Kosten. Sie aktivieren diese Tags im AWS Billing and Cost Management-Dashboard. AWS verwendet die Tags, um Ihre Kosten zu kategorisieren und Ihnen einen monatlichen Kostenzuordnungsbericht zu liefern. Weitere Informationen finden Sie unter Verwendung von Tags zur Kostenzuordnung im Benutzerhandbuch zu AWS Billing.
- Steuern Sie den Zugriff auf IhreAWS Ressourcen mitAWS Identity and Access Management (IAM). Weitere Informationen finden SieAttributbasierte Zugriffskontrolle (ABAC) mit QLDB in diesem Entwicklerhandbuch und unter Steuern des Zugriffs mithilfe von IAM-Tags im IAM-Benutzerhandbuch.

Tipps zur Verwendung von Tags finden Sie unter AWS-Tagging-Strategien im AWS-Antworten-Blog.

In den folgenden Abschnitten erhalten Sie weitere Informationen zu Tags für Amazon QLDB.

Themen

- Unterstützte Ressourcen in Amazon QLDB
- Konventionen für die Tag-Benennung und -Verwendung

- Verwalten von Tags
- Markieren von Ressourcen bei der Erstellung

Unterstützte Ressourcen in Amazon QLDB

Die folgenden Ressourcen in Amazon QLDB unterstützen die Markierung:

- Ledger
- Tabelle
- Journal-Stream

Weitere Informationen zum Hinzufügen und Verwalten von Tags finden Sie unter Verwalten von Tags.

Konventionen für die Tag-Benennung und -Verwendung

Die folgenden grundlegenden Konventionen für die Benennung und Verwendung gelten für die Verwendung von Tags mit Amazon QLDB Ressourcen:

- Jede Ressource kann maximal 50 Tags haben.
- Jeder Tag muss für jede Ressource eindeutig sein. Jeder Tag kann nur einen Wert haben.
- Die maximale Länge des Tag-Schlüssels beträgt 128 Unicode-Zeichen in UTF-8.
- Die maximale Länge des Tag-Wertes beträgt 256 Unicode-Zeichen in UTF-8.
- Erlaubte Zeichen sind Buchstaben, Ziffern und Leerzeichen, die in UTF-8 darstellbar sind, sowie die folgenden Zeichen: . : + = @ _ / - (Bindestrich).
- Bei Tag-Schlüsseln und -Werten muss die Groß- und Kleinschreibung beachtet werden. Eine bewährte Methode besteht darin, sich für eine einheitliche Schreibweise der Tag-Benennungen zu entscheiden und diese Strategie für alle Ressourcentypen umzusetzen. Entscheiden Sie sich beispielsweise für `Costcenter`, `costcenter` oder `CostCenter` und verwenden Sie diese Konvention für alle Tags. Vermeiden Sie die Verwendung von ähnlichen Tags mit uneinheitlicher Fallunterscheidung.
- Das Präfix `aws:` ist zur Verwendung in AWS reserviert. Sie können den Schlüssel oder Wert eines Tags nicht bearbeiten oder löschen, wenn das Tag über einen Tag-Schlüssel mit dem Präfix `aws:` verfügt. Tags mit diesem Präfix werden nicht als Ihre Tags pro Ressourcenlimit angerechnet.

Verwalten von Tags

Tags bestehen aus den Eigenschaften `Key` und `Value` für eine Ressource. In der Amazon QLDB ConsoleAWS CLI, oder der QLDB-API können Sie Werte für diese Eigenschaften hinzufügen, bearbeiten oder löschen. Sie können auch den AWS Resource Groups-Tag-Editor verwenden, um Tags zu verwalten.

Informationen zum Arbeiten mit Tags finden Sie in den folgenden API-Operationen:

- ListTagsForResourcein der Amazon QLDB-API-Referenz
- TagResourcein der Amazon QLDB-API-Referenz
- UntagResourcein der Amazon QLDB-API-Referenz

So verwenden Sie das QLDB-Tagging-Panel (Konsole)

1. Melden Sie sich bei derAWS Management Console an und öffnen Sie die Amazon QLDB Console unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich Ledgers aus.
3. Wählen Sie in der Liste Ledgers (Ledger) den Namen des Ledgers aus, dessen Tags Sie verwalten möchten.
4. Suchen Sie auf der Detailseite des Ledger die Karte Tags und wählen Sie Manage tags (Tags bearbeiten) aus.
5. Auf der Seite Mange tags (Tags verwalten) können Sie Tags für Ihren Ledger ggf. hinzufügen, bearbeiten oder entfernen. Wenn die Tag-Schlüssel und -Werte Ihren Wünschen entsprechen, wählen Sie Save (Speichern) aus.

Markieren von Ressourcen bei der Erstellung

Für QLDB-Ressourcen, die Tagging unterstützen, können Sie Tags definieren, während Sie die Ressource erstellen, indem Sie dieAWS Management ConsoleAWS CLI, die oder die QLDB-API verwenden. Indem Sie Ressourcen zum Erstellungszeitpunkt markieren, müssen Sie anschließend keine benutzerdefinierten Markierungs-Skripts ausführen.

Nachdem eine Ressource markiert wurde, können Sie den Zugriff auf die Ressource anhand dieser Tags steuern. Sie können beispielsweise nur Tabellenressourcen vollen Zugriff gewähren, die über

ein bestimmtes Tag verfügen. Ein Beispiel für eine JSON-Richtlinie finden Sie unterVollzugriff auf alle Aktionen basierend auf Tabellen-Tags.

> **Note**
>
> Tabellen- und Stream-Ressourcen erben nicht die Tags ihrer Root-Ledger-Ressource.

Sie können Tabellen-Tags auch definieren, indem Sie sie in einer`CREATE TABLE` PartiQL-Anweisung angeben. Weitere Informationen hierzu finden Sie unter Markieren von Tabellen.

Sicherheit in Amazon QLDB

Cloud-Sicherheit bei AWS hat höchste Priorität. Als - AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die entwickelt wurde, um die Anforderungen der sicherheitssensibelsten Organisationen zu erfüllen.

Sicherheit ist eine geteilte Verantwortung zwischen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud – AWS ist für den Schutz der Infrastruktur verantwortlich, die AWS-Services in der ausgeführt wird AWS Cloud. stellt Ihnen AWS außerdem Services bereit, die Sie sicher nutzen können. Auditoren von Drittanbietern testen und überprüfen die Effektivität unserer Sicherheitsmaßnahmen im Rahmen der AWS -Compliance-Programme regelmäßig. Weitere Informationen zu den Compliance-Programmen, die für Amazon QLDB gelten, finden Sie unter AWS Im Rahmen des Compliance-Programms zugelassene -Services.
- Sicherheit in der Cloud – Ihre Verantwortung wird durch die bestimmt AWS-Service , die Sie verwenden. Sie sind auch für andere Faktoren verantwortlich, einschließlich der Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der geteilten Verantwortung bei der Verwendung von QLDB einsetzen können. Die folgenden Themen zeigen Ihnen, wie Sie QLDB zur Erfüllung Ihrer Sicherheits- und Compliance-Ziele konfigurieren. Sie erfahren auch, wie Sie andere verwenden AWS-Services , die Ihnen bei der Überwachung und Sicherung Ihrer QLDB-Ressourcen helfen.

Themen

Datenschutz in Amazon QLDB

Das AWS Modell der geteilten VerantwortungModell gilt für den Datenschutz in Amazon QLDB. Wie in diesem Modell beschrieben, AWS ist für den Schutz der globalen Infrastruktur verantwortlich, die alle ausführt AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir Ihnen, -Anmeldeinformationen zu schützen AWS-Konto und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einzurichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor Authentifizierung (MFA).
- Verwenden Sie SSL/TLS für die Kommunikation mit - AWS Ressourcen. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API- und Benutzeraktivitätsprotokollierung mit ein AWS CloudTrail.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen Standardsicherheitskontrollen in AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff auf AWS über eine Befehlszeilenschnittstelle oder eine API FIPS-140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit QLDB oder anderen AWS-Services über die Konsole, API AWS CLIoder AWS SDKs arbeiten. Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

> **Note**
>
> Diese Anleitung zur Vermeidung von Tags oder Freiformfeldern für sensible Informationen bezieht sich auf die Metadaten einer QLDB-Ledger-Ressource und nicht auf Daten, die im Ledger gespeichert sind. Ihre in einer QLDB-Ledger-Ressource gespeicherten Daten werden nicht für Fakturierungs- oder Diagnoseprotokolle verwendet.

Themen

Verschlüsselung im Ruhezustand in Amazon QLDB

Alle in Amazon QLDB gespeicherten Daten werden im Ruhezustand standardmäßig vollständig verschlüsselt. Die QLDB-Verschlüsselung im Ruhezustand verbessert die Sicherheit, indem alle Ledger-Daten im Ruhezustand mit Verschlüsselungsschlüsseln in AWS Key Management Service () verschlüsselt werdenAWS KMS. Diese Funktionalität trägt zur Verringerung des Betriebsaufwands und der Komplexität bei, die mit dem Schutz sensibler Daten einhergeht. Mit der Verschlüsselung im Ruhezustand können Sie sicherheitsrelevante Ledger-Anwendungen erstellen, die strenge Verschlüsselungs-Compliance und regulatorische Anforderungen erfüllen.

Die Verschlüsselung im Ruhezustand ist in integriert AWS KMS , um den Verschlüsselungsschlüssel zu verwalten, der zum Schutz Ihrer QLDB-Ledger verwendet wird. Weitere Informationen zu finden Sie unter AWS KMS-AWS Key Management Service Konzepte im AWS Key Management Service - Entwicklerhandbuch.

In QLDB können Sie den Typ von AWS KMS key für jede Ledger-Ressource angeben. Wenn Sie ein neues Ledger erstellen oder ein vorhandenes Ledger aktualisieren, können Sie einen der folgenden KMS-Schlüsseltypen auswählen, um Ihre Ledger-Daten zu schützen:

- AWS-eigener Schlüssel – Der Standardverschlüsselungstyp. Der Schlüssel gehört QLDB (keine zusätzlichen Gebühren).
- Kundenverwalteter Schlüssel – Der Schlüssel wird in Ihrem gespeichert AWS-Konto und von Ihnen erstellt, besessen und verwaltet. Sie haben die volle Kontrolle über den Schlüssel (es fallenAWS KMS Gebühren an).

> **Note**
>
> Amazon QLDB hat AWS KMS keys am 22. Juli 2021 Support für kundenverwaltete eingeführt. Alle Ledger, die vor dem Start erstellt wurden, sind AWS-eigene Schlüssel standardmäßig geschützt, sind aber derzeit nicht für die Verschlüsselung im Ruhezustand mit vom Kunden verwalteten Schlüsseln berechtigt.
> Sie können die Erstellungszeit Ihres Ledgers in der QLDB-Konsole anzeigen.

Wenn Sie auf einen Ledger zugreifen, entschlüsselt QLDB die Daten transparent. Sie können jederzeit zwischen dem AWS-eigener Schlüssel und dem vom Kunden verwalteten Schlüssel wechseln. Sie müssen keinen Code oder Anwendungen ändern, um verschlüsselte Daten zu verwenden oder zu verwalten.

Sie können einen Verschlüsselungsschlüssel angeben, wenn Sie einen neuen Ledger erstellen, oder den Verschlüsselungsschlüssel für einen vorhandenen Ledger ändern, indem Sie die AWS Management Console, die QLDB-API oder die AWS Command Line Interface () verwendenAWS CLI. Weitere Informationen finden Sie unter Verwenden von kundenverwalteten Schlüsseln in Amazon QLDB.

> **Note**
>
> Standardmäßig aktiviert Amazon QLDB automatisch die Verschlüsselung im Ruhezustand mit AWS-eigene Schlüssel ohne zusätzliche Kosten. Für die Verwendung eines vom Kunden verwalteten Schlüssels fallen jedoch AWS KMS Gebühren an. Informationen zu Preisen finden Sie unter AWS Key Management Service -Preise.
> QLDB-Verschlüsselung im Ruhezustand ist in allen verfügbar AWS-Regionen , in denen QLDB verfügbar ist.

Themen

- Verschlüsselung im Ruhezustand: Funktionsweise in Amazon QLDB
- Verwenden von kundenverwalteten Schlüsseln in Amazon QLDB

Verschlüsselung im Ruhezustand: Funktionsweise in Amazon QLDB

QLDB-Verschlüsselung im Ruhezustand verschlüsselt Ihre Daten mit dem 256-Bit Advanced Encryption Standard (AES-256). Dies trägt dazu bei, Ihre Daten vor unbefugtem Zugriff auf den zugrunde liegenden Speicher zu schützen. Alle in QLDB-Ledgern gespeicherten Daten werden im Ruhezustand standardmäßig verschlüsselt. Die serverseitige Verschlüsselung ist transparent, was bedeutet, dass keine Änderungen an Anwendungen erforderlich sind.

Die Verschlüsselung im Ruhezustand ist in AWS Key Management Service (AWS KMS) integriert, um den Verschlüsselungsschlüssel zu verwalten, der zum Schutz Ihrer QLDB-Ledger verwendet wird. Wenn Sie ein neues Ledger erstellen oder ein vorhandenes AWS KMS Ledger aktualisieren, können Sie einen der folgenden Schlüsseltypen auswählen:

- AWS-eigener Schlüssel – Der Standardverschlüsselungstyp. Der Schlüssel gehört QLDB (keine zusätzlichen Gebühren).
- Kundenverwalteter Schlüssel – Der Schlüssel wird in Ihrem gespeichert AWS-Konto und von Ihnen erstellt, besessen und verwaltet. Sie haben die volle Kontrolle über den Schlüssel (es fallenAWS KMS Gebühren an).

Themen

AWS-eigener Schlüssel

AWS-eigene Schlüssel werden nicht in Ihrem gespeichert AWS-Konto. Sie sind Teil einer Sammlung von KMS-Schlüsseln, die AWS besitzt und für die Verwendung in mehreren verwaltet AWS-Konten. AWS-Services kann AWS-eigene Schlüssel zum Schutz Ihrer Daten verwenden.

Sie müssen nicht erstellen oder verwalten AWS-eigene Schlüssel. Sie können jedoch nicht anzeigen oder verfolgen AWS-eigene Schlüsseloder deren Verwendung überprüfen. Ihnen wird keine monatliche Gebühr oder Nutzungsgebühr für berechnet AWS-eigene Schlüsselund sie werden nicht auf die AWS KMS Kontingente für Ihr Konto angerechnet.

Weitere Informationen finden Sie unter AWS-eigene Schlüssel im AWS Key Management Service - Entwicklerhandbuch.

Kundenverwalteter Schlüssel

Kundenverwaltete Schlüssel sind KMS-Schlüssel in Ihrem , AWS-Konto die Sie erstellen, besitzen und verwalten. Sie haben die volle Kontrolle über diese KMS-Schlüssel. QLDB unterstützt nur KMS-Schlüssel mit symmetrischer Verschlüsselung.

Verwenden Sie einen kundenverwalteten KMS-Schlüssel, um die folgenden Funktionen zu erhalten:

- Festlegen und Verwalten von Schlüsselrichtlinien, IAM-Richtlinien und Erteilungen zur Steuerung des Zugriffs auf den Schlüssel
- Aktivieren und Deaktivieren des Schlüssels
- Rotieren von kryptografischem Material für den Schlüssel
- Erstellen von Schlüssel-Tags und Aliassen
- Planen des Schlüssels zum Löschen
- Importieren Ihres eigenen Schlüsselmaterials oder Verwenden eines benutzerdefinierten Schlüsselspeichers, den Sie besitzen und verwalten
- Verwenden von AWS CloudTrail und Amazon CloudWatch Logs zum Nachverfolgen der Anforderungen, die QLDB AWS KMS in Ihrem Namen an sendet

Weitere Informationen finden Sie unter Kundenverwaltete Schlüssel im AWS Key Management Service Entwicklerhandbuch.

Für kundenverwaltete Schlüssel fällt für jeden API-Aufruf eine Gebühr an, und für diese KMS-Schlüssel gelten AWS KMS Kontingente. Weitere Informationen finden Sie unter AWS KMS Ressourcen- oder Anforderungskontingente.

Wenn Sie einen vom Kunden verwalteten Schlüssel als KMS-Schlüssel für einen Ledger angeben, werden alle Ledger-Daten sowohl im Journalspeicher als auch im indizierten Speicher mit demselben vom Kunden verwalteten Schlüssel geschützt.

Unzugängliche kundenverwaltete Schlüssel

Wenn Sie Ihren kundenverwalteten Schlüssel deaktivieren, den Schlüssel zum Löschen planen oder die Erteilungen für den Schlüssel widerrufen, wird der Status Ihrer Ledger-Verschlüsselung zu `KMS_KEY_INACCESSIBLE`. In diesem Zustand ist der Ledger beeinträchtigt und akzeptiert keine

Lese- oder Schreibanforderungen. Ein unzugänglicher Schlüssel verhindert, dass alle Benutzer und der QLDB-Service Daten ver- oder entschlüsseln und Lese- und Schreibvorgänge im Ledger ausführen. QLDB muss Zugriff auf Ihren KMS-Schlüssel haben, um sicherzustellen, dass Sie weiterhin auf Ihr Ledger zugreifen können, und um Datenverlust zu verhindern.

> **Important**
>
> Ein beeinträchtigtes Ledger kehrt automatisch in einen aktiven Zustand zurück, nachdem Sie die Erteilungen für den Schlüssel wiederhergestellt haben oder nachdem Sie den deaktivierten Schlüssel wieder aktiviert haben.
> Das Löschen eines kundenverwalteten Schlüssels ist jedoch irreversibel. Nachdem ein Schlüssel gelöscht wurde, können Sie nicht mehr auf die Ledger zugreifen, die mit diesem Schlüssel geschützt sind, und die Daten können nicht dauerhaft wiederhergestellt werden.

Um den Verschlüsselungsstatus eines Ledgers zu überprüfen, verwenden Sie die - AWS Management Console oder die DescribeLedger-API-Operation.

So verwendet Amazon QLDB Erteilungen in AWS KMS

QLDB benötigt Erteilungen, um Ihren vom Kunden verwalteten Schlüssel zu verwenden. Wenn Sie ein Ledger erstellen, das mit einem vom Kunden verwalteten Schlüssel geschützt ist, erstellt QLDB in Ihrem Namen Erteilungen, indem CreateGrant Anfragen an gesendet werden AWS KMS. Erteilungen in AWS KMS werden verwendet, um QLDB Zugriff auf einen KMS-Schlüssel in einem Kunden zu gewähren AWS-Konto. Weitere Informationen finden Sie unter Verwenden von Erteilungen im AWS Key Management Service -Entwicklerhandbuch.

QLDB benötigt die Erteilungen, um Ihren vom Kunden verwalteten Schlüssel für die folgenden AWS KMS Vorgänge zu verwenden:

- DescribeKey – Überprüfen Sie, ob der angegebene KMS-Schlüssel mit symmetrischer Verschlüsselung gültig ist.
- GenerateDataKey – Generieren Sie einen eindeutigen symmetrischen Datenschlüssel, den QLDB verwendet, um Daten im Ruhezustand in Ihrem Ledger zu verschlüsseln.
- Entschlüsseln – Entschlüsseln Sie den Datenschlüssel, der mit Ihrem vom Kunden verwalteten Schlüssel verschlüsselt wurde.
- Verschlüsseln – Verschlüsseln Sie Klartext mit Ihrem vom Kunden verwalteten Schlüssel in Geheimtext.

Sie können eine Erteilung widerrufen, um den Zugriff des Services auf den vom Kunden verwalteten Schlüssel jederzeit zu entfernen. Wenn Sie dies tun, ist der Schlüssel nicht zugänglich, und QLDB verliert den Zugriff auf Ledger-Daten, die durch den vom Kunden verwalteten Schlüssel geschützt sind. In diesem Zustand ist der Ledger beeinträchtigt und akzeptiert keine Lese- oder Schreibanforderungen, bis Sie die Erteilungen für den Schlüssel wiederherstellen.

Wiederherstellen von Erteilungen in AWS KMS

Um Erteilungen für einen vom Kunden verwalteten Schlüssel wiederherzustellen und den Zugriff auf einen Ledger in QLDB wiederherzustellen, können Sie den Ledger aktualisieren und denselben KMS-Schlüssel angeben. Anweisungen finden Sie unter Aktualisieren des eines vorhandenen AWS KMS key Ledgers.

Überlegungen zur Verschlüsselung im Ruhezustand

Beachten Sie Folgendes, wenn Sie die Verschlüsselung im Ruhezustand in QLDB verwenden:

- Die serverseitige Verschlüsselung im Ruhezustand ist standardmäßig für alle QLDB-Ledger-Daten aktiviert und kann nicht deaktiviert werden. Sie können nicht nur eine Teilmenge von Daten in einem Ledger verschlüsseln.
- Die Verschlüsselung ruhender Daten verschlüsselt Daten nur, während sie auf persistenten Speichermedien statisch (ruhende Daten) sind. Wenn die Datensicherheit bei Daten während der Übertragung oder bei der Verwendung von Daten ein Problem darstellt, müssen Sie möglicherweise wie folgt zusätzliche Maßnahmen ergreifen:
  - Daten während der Übertragung: Alle Ihre Daten in QLDB werden während der Übertragung verschlüsselt. Standardmäßig verwenden die Kommunikation zu und von QLDB das HTTPS-Protokoll, das den Netzwerkverkehr mithilfe der Secure Sockets Layer (SSL)/Transport Layer Security (TLS)-Verschlüsselung schützt.
  - Verwendete Daten: Schützen Sie Ihre Daten, bevor Sie sie mithilfe der clientseitigen Verschlüsselung an QLDB senden.

Um zu erfahren, wie Sie vom Kunden verwaltete Schlüssel für Ledger implementieren, fahren Sie mit fortVerwenden von kundenverwalteten Schlüsseln in Amazon QLDB.

Verwenden von kundenverwalteten Schlüsseln in Amazon QLDB

Sie können die AWS Management Console, die AWS Command Line Interface (AWS CLI) oder die QLDB-API verwenden, um die AWS KMS key für neue Ledger und vorhandene Ledger in Amazon

QLDB anzugeben. In den folgenden Themen wird beschrieben, wie Sie die Nutzung Ihrer vom Kunden verwalteten Schlüssel in QLDB verwalten und überwachen.

Themen

Voraussetzungen

Bevor Sie einen QLDB-Ledger mit einem vom Kunden verwalteten Schlüssel schützen können, müssen Sie zuerst den Schlüssel in AWS Key Management Service () erstellenAWS KMS. Sie müssen auch eine Schlüsselrichtlinie angeben, die es QLDB ermöglicht, in diesem AWS KMS key in Ihrem Namen Erteilungen zu erstellen.

Erstellen eines vom Kunden verwalteten Schlüssels

Um einen kundenverwalteten Schlüssel zu erstellen, führen Sie die Schritte unter Erstellen von KMS-Schlüsseln mit symmetrischer Verschlüsselung im AWS Key Management Service - Entwicklerhandbuch aus. QLDB unterstützt keine asymmetrischen Schlüssel.

Einstellen einer Schlüsselrichtlinie

Schlüsselrichtlinien sind die primäre Möglichkeit, den Zugriff auf vom Kunden verwaltete Schlüssel in zu steuern AWS KMS. Jeder vom Kunden verwaltete Schlüssel muss genau eine Schlüsselrichtlinie haben. Die Anweisungen im Schlüsselrichtliniendokument legen fest, wer über eine Berechtigung zur Verwendung des KMS-Schlüssels verfügt, und wie diese Verwendung erfolgen kann. Weitere Informationen finden Sie unter Verwenden von Schlüsselrichtlinien in AWS KMS.

Sie können eine Schlüsselrichtlinie angeben, wenn Sie Ihren kundenverwalteten Schlüssel erstellen. Informationen zum Ändern einer Schlüsselrichtlinie für einen vorhandenen kundenverwalteten Schlüssel finden Sie unter Ändern einer Schlüsselrichtlinie.

Damit QLDB Ihren vom Kunden verwalteten Schlüssel verwenden kann, muss die Schlüsselrichtlinie Berechtigungen für die folgenden AWS KMS Aktionen enthalten:

- kms:CreateGrant – Fügt einem vom Kunden verwalteten Schlüssel eine Berechtigung hinzu. Gewährt Kontrollzugriff auf einen angegebenen KMS-Schlüssel.

Wenn Sie einen Ledger mit einem bestimmten kundenverwalteten Schlüssel erstellen oder aktualisieren, erstellt QLDB Erteilungen, die den Zugriff auf die benötigten Erteilungsvorgänge ermöglichen. Zu den Erteilungsvorgängen gehören die folgenden:

- GenerateDataKey
- Decrypt
- Encrypt

- kms:DescribeKey – Gibt detaillierte Informationen zu einem vom Kunden verwalteten Schlüssel zurück. QLDB verwendet diese Informationen, um den Schlüssel zu validieren.

Beispiel für eine Schlüsselrichtlinie

Im Folgenden finden Sie ein Beispiel für eine Schlüsselrichtlinie, die Sie für QLDB verwenden können. Diese Richtlinie ermöglicht es Prinzipalen, die berechtigt sind, QLDB vom Konto aus zu verwenden`111122223333`, die - `DescribeKey` und -`CreateGrant`Operationen für die Ressource aufzurufen`arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`.

Um diese Richtlinie zu verwenden, ersetzen Sie *`us-east-1`*, *`111122223333`* und *`1234abcd-12ab-34cd-56ef-1234567890ab`* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid" : "Allow access to principals authorized to use Amazon QLDB",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "*"
            },
            "Action" : [
                "kms:DescribeKey",
                "kms:CreateGrant"
            ],
            "Resource" : "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "Condition" : {
                "StringEquals" : {
                    "kms:ViaService" : "qldb.us-east-1.amazonaws.com",
                    "kms:CallerAccount" : "111122223333"
```

```
                }
            }
        }
    ]
}
```

Angeben der AWS KMS key für einen neuen Ledger

Gehen Sie wie folgt vor, um einen KMS-Schlüssel anzugeben, wenn Sie mithilfe der QLDB-Konsole oder der einen neuen Ledger erstellen AWS CLI.

Sie können einen vom Kunden verwalteten Schlüssel angeben, indem Sie eine ID, einen Alias oder einen Amazon-Ressourcennamen (ARN) verwenden. Weitere Informationen finden Sie unter Schlüsselkennungen (KeyId) im AWS Key Management Service Entwicklerhandbuch für .

> **Note**
>
> Regionsübergreifende Schlüssel werden nicht unterstützt. Der angegebene KMS-Schlüssel muss sich in derselben AWS-Region wie Ihr Ledger befinden.

Erstellen eines Ledgers (Konsole)

1. Melden Sie sich bei der an AWS Management Consoleund öffnen Sie die Amazon-QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie Create Ledger (Ledger erstellen) aus.
3. Gehen Sie auf der Seite Create Ledger (Ledger erstellen) wie folgt vor:

   - Ledger-Informationen – Geben Sie einen Ledger-Namen ein, der unter allen Ledgern im aktuellen AWS-Konto und in der Region eindeutig ist.
   - Berechtigungsmodus – Wählen Sie einen Berechtigungsmodus aus, der dem Ledger zugewiesen werden soll:
     - Alle zulassen
     - Standard (empfohlen)
   - Daten im Ruhezustand verschlüsseln – Wählen Sie den KMS-Schlüsseltyp aus, der für die Verschlüsselung im Ruhezustand verwendet werden soll:

- Verwenden eines AWS eigenen KMS-Schlüssels – Verwenden Sie einen KMS-Schlüssel, der AWS in Ihrem Namen Eigentum von ist und von verwaltet wird. Dies ist die Standardoption und erfordert keine zusätzliche Einrichtung.
- Wählen Sie einen anderen AWS KMS Schlüssel – Verwenden Sie einen KMS-Schlüssel mit symmetrischer Verschlüsselung in Ihrem Konto, den Sie erstellen, besitzen und verwalten.

  Um einen neuen Schlüssel mithilfe der AWS KMS Konsole zu erstellen, wählen Sie Erstellen eines - AWS KMS Schlüssels aus. Weitere Informationen finden Sie unter Erstellen symmetrischer KMS-Verschlüsselungsschlüssel im AWS Key Management Service - Entwicklerhandbuch.

  Um einen vorhandenen KMS-Schlüssel zu verwenden, wählen Sie einen aus der Dropdown-Liste aus oder geben Sie einen KMS-Schlüssel-ARN an.

4. Wenn Sie die gewünschten Einstellungen vorgenommen haben, wählen Sie Create ledger (Ledger erstellen) aus.

   Sie können auf Ihr QLDB-Ledger zugreifen, wenn sein Status Aktiv wird. Dies kann mehrere Minuten dauern.

Erstellen eines Ledgers (AWS CLI)

Verwenden Sie die AWS CLI , um einen Ledger in QLDB mit dem Standard- AWS-eigener Schlüssel oder einem vom Kunden verwalteten Schlüssel zu erstellen.

Example – So erstellen Sie ein Ledger mit dem Standard AWS-eigener Schlüssel

```
aws qldb create-ledger --name my-example-ledger --permissions-mode STANDARD
```

Example – So erstellen Sie ein Ledger mit einem vom Kunden verwalteten Schlüssel

```
aws qldb create-ledger \
    --name my-example-ledger \
    --permissions-mode STANDARD \
    --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Aktualisieren des eines vorhandenen AWS KMS key Ledgers

Sie können auch die QLDB-Konsole oder die verwenden AWS CLI , um den KMS-Schlüssel eines vorhandenen Ledgers jederzeit auf einen AWS-eigener Schlüssel oder einen vom Kunden verwalteten Schlüssel zu aktualisieren.

> **Note**
>
> Amazon QLDB hat AWS KMS keys am 22. Juli 2021 Support für kundenverwaltete eingeführt. Alle Ledger, die vor dem Start erstellt wurden, sind AWS-eigene Schlüssel standardmäßig geschützt, sind aber derzeit nicht für die Verschlüsselung im Ruhezustand mit vom Kunden verwalteten Schlüsseln berechtigt.
> Sie können die Erstellungszeit Ihres Ledgers in der QLDB-Konsole anzeigen.

Schlüsseländerungen in QLDB sind asynchron. Der Ledger ist vollständig zugänglich, ohne dass sich dies auf die Leistung auswirkt, während die Schlüsseländerung verarbeitet wird. Die Zeit, die zum Aktualisieren eines Schlüssels benötigt wird, variiert je nach Ledger-Größe.

Sie können einen vom Kunden verwalteten Schlüssel angeben, indem Sie eine ID, einen Alias oder einen Amazon-Ressourcennamen (ARN) verwenden. Weitere Informationen finden Sie unter Schlüsselkennungen (KeyId) im AWS Key Management Service Entwicklerhandbuch für .

> **Note**
>
> Regionsübergreifende Schlüssel werden nicht unterstützt. Der angegebene KMS-Schlüssel muss sich in derselben AWS-Region wie Ihr Ledger befinden.

Aktualisieren eines Ledgers (Konsole)

1. Melden Sie sich bei der an AWS Management Consoleund öffnen Sie die Amazon-QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich Ledgers aus.
3. Wählen Sie in der Liste der Ledger das Ledger aus, das Sie aktualisieren möchten, und wählen Sie dann Ledger bearbeiten aus.
4. Wählen Sie auf der Seite Ledger bearbeiten den Typ des KMS-Schlüssels aus, der für die Verschlüsselung im Ruhezustand verwendet werden soll:

- Verwenden eines AWS eigenen KMS-Schlüssels – Verwenden Sie einen KMS-Schlüssel, der AWS in Ihrem Namen Eigentum von ist und von verwaltet wird. Dies ist die Standardoption und erfordert keine zusätzliche Einrichtung.
- Wählen Sie einen anderen AWS KMS Schlüssel – Verwenden Sie einen KMS-Schlüssel mit symmetrischer Verschlüsselung in Ihrem Konto, den Sie erstellen, besitzen und verwalten.

  Um einen neuen Schlüssel mithilfe der AWS KMS Konsole zu erstellen, wählen Sie Erstellen eines - AWS KMS Schlüssels aus. Weitere Informationen finden Sie unter Erstellen symmetrischer KMS-Verschlüsselungsschlüssel im AWS Key Management Service - Entwicklerhandbuch.

  Um einen vorhandenen KMS-Schlüssel zu verwenden, wählen Sie einen aus der Dropdown-Liste aus oder geben Sie einen KMS-Schlüssel-ARN an.

5. Wählen Sie Änderungen bestätigen.

Aktualisieren eines Ledgers (AWS CLI)

Verwenden Sie die AWS CLI , um ein vorhandenes Ledger in QLDB mit dem Standard- AWS-eigener Schlüssel oder einem vom Kunden verwalteten Schlüssel zu aktualisieren.

Example – So aktualisieren Sie einen Ledger mit dem Standard AWS-eigener Schlüssel

```
aws qldb update-ledger --name my-example-ledger --kms-key AWS_OWNED_KMS_KEY
```

Example – So aktualisieren Sie einen Ledger mit einem vom Kunden verwalteten Schlüssel

```
aws qldb update-ledger \
    --name my-example-ledger \
    --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Überwachung Ihrer AWS KMS keys

Wenn Sie einen vom Kunden verwalteten Schlüssel zum Schutz Ihrer Amazon-QLDB-Ledger verwenden, können Sie AWS CloudTrail oder Amazon CloudWatch Logs verwenden, um die Anforderungen zu verfolgen, die QLDB AWS KMS in Ihrem Namen an sendet. Weitere Informationen finden Sie unter Überwachung AWS KMS keys im AWS Key Management Service - Entwicklerhandbuch.

Die folgenden Beispiele sind CloudTrail Protokolleinträge für die Operationen `CreateGrant`, `GenerateDataKey`, `DecryptEncrypt`, und `DescribeKey`.

CreateGrant

Wenn Sie einen vom Kunden verwalteten Schlüssel zum Schutz Ihres Ledgers angeben, sendet QLDB AWS KMS in Ihrem Namen `CreateGrant` Anfragen an , um den Zugriff auf Ihren KMS-Schlüssel zu ermöglichen. Darüber hinaus verwendet QLDB die -`RetireGrant`Operation, um Erteilungen zu entfernen, wenn Sie einen Ledger löschen.

Die von QLDB erstellten Erteilungen sind spezifisch für einen Ledger. Der Prinzipal in der `CreateGrant` Anforderung ist der Benutzer, der die Tabelle erstellt hat.

Das Ereignis, das die `CreateGrant`-Operation aufzeichnet, ähnelt dem folgenden Beispielereignis. Zu den Parametern gehören der Amazon-Ressourcenname (ARN) des vom Kunden verwalteten Schlüssels, der Empfänger-Prinzipal und der ausscheidende Prinzipal (der QLDB-Service) sowie die Operationen, die die Erteilung abdeckt.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:sample-user",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/sample-user",
        "accountId": "111122223333",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AKIAIOSFODNN7EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-06-04T21:37:11Z"
            }
        },
        "invokedBy": "qldb.amazonaws.com"
    },
```

```
    "eventTime": "2021-06-04T21:40:00Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "qldb.amazonaws.com",
    "userAgent": "qldb.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "granteePrincipal": "qldb.us-west-2.amazonaws.com",
        "operations": [
            "DescribeKey",
            "GenerateDataKey",
            "Decrypt",
            "Encrypt"
        ],
        "retiringPrincipal": "qldb.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "b3c83f999187ccc0979ef2ff86a1572237b6bba309c0ebce098c34761f86038a"
    },
    "requestID": "e99188d7-3b82-424e-b63e-e086d848ed60",
    "eventID": "88dc7ba5-4952-4d36-9ca8-9ab5d9598bab",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

GenerateDataKey

Wenn Sie einen vom Kunden verwalteten Schlüssel zum Schutz Ihres Ledgers angeben, erstellt QLDB einen eindeutigen Datenschlüssel. Es sendet eine `GenerateDataKey` Anforderung an AWS KMS , die den vom Kunden verwalteten Schlüssel für das Ledger angibt.

Das Ereignis, das die `GenerateDataKey`-Operation aufzeichnet, ähnelt dem folgenden Beispielereignis. Der Benutzer ist das QLDB-Servicekonto. Die Parameter umfassen den ARN des vom Kunden verwalteten Schlüssels, einen Datenschlüsselbezeichner, der eine Länge von 32 Byte erfordert, und den Verschlüsselungskontext, der den internen Schlüsselhierarchieknoten identifiziert.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "qldb.amazonaws.com"
    },
    "eventTime": "2021-06-04T21:40:01Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "qldb.amazonaws.com",
    "userAgent": "qldb.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "numberOfBytes": 32,
        "encryptionContext": {
            "key-hierarchy-node-id": "LY4HWMnkeZWKYi6MlitVJC",
            "key-hierarchy-node-version": "1"
        }
    },
    "responseElements": null,
    "requestID": "786977c9-e77c-467a-bff5-9ad5124a4462",
    "eventID": "b3f082cb-3e75-454e-bf0a-64be13075436",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
```

```
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "26688de5-0b1c-43d3-bc4f-a18029b08446"
}
```

Decrypt

Wenn Sie auf einen Ledger zugreifen, ruft QLDB die -`Decrypt`Operation auf, um den gespeicherten Datenschlüssel des Ledgers zu entschlüsseln, damit er auf die verschlüsselten Daten im Ledger zugreifen kann.

Das Ereignis, das die `Decrypt`-Operation aufzeichnet, ähnelt dem folgenden Beispielereignis. Der Benutzer ist das QLDB-Servicekonto. Die Parameter umfassen den ARN des vom Kunden verwalteten Schlüssels und den Verschlüsselungskontext, der den internen Schlüsselhierarchieknoten identifiziert.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "qldb.amazonaws.com"
    },
    "eventTime": "2021-06-04T21:40:56Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "qldb.amazonaws.com",
    "userAgent": "qldb.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "encryptionContext": {
            "key-hierarchy-node-id": "LY4HWMnkeZWKYi6MlitVJC",
            "key-hierarchy-node-version": "1"
        }
    },
    "responseElements": null,
    "requestID": "28f2dd18-3cc1-4fe2-82f7-5154f4933ebf",
```

```
    "eventID": "603ad5d4-4744-4505-9c21-bd4a6cbd4b20",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "7b6ce3e3-a764-42ec-8f90-5418c97ec411"
}
```

Encrypt

QLDB ruft die -`Encrypt`Operation auf, um Klartext mithilfe Ihres vom Kunden verwalteten Schlüssels in Geheimtext zu verschlüsseln.

Das Ereignis, das die `Encrypt`-Operation aufzeichnet, ähnelt dem folgenden Beispielereignis. Der Benutzer ist das QLDB-Servicekonto. Die Parameter umfassen den ARN des vom Kunden verwalteten Schlüssels und den Verschlüsselungskontext, der die interne eindeutige ID des Ledgers angibt.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "qldb.amazonaws.com"
    },
    "eventTime": "2021-06-04T21:40:01Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Encrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "qldb.amazonaws.com",
    "userAgent": "qldb.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionContext": {
```

```
            "LedgerId": "F6qRNziJLUXA4Vy2ZUv8YY"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "b2daca7d-4606-4302-a2d7-5b3c8d30c64d",
    "eventID": "b8aace05-2e37-4fed-ae6f-a45a1c6098df",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "ce420ab0-288e-4b4f-aee8-541e18a28aa5"
}
```

DescribeKey

QLDB ruft die -`DescribeKey`Operation auf, um festzustellen, ob der von Ihnen angegebene KMS-Schlüssel in der AWS-Konto Region und vorhanden ist.

Das Ereignis, das die `DescribeKey`-Operation aufzeichnet, ähnelt dem folgenden Beispielereignis. Der Prinzipal ist der Benutzer in Ihrem AWS-Konto , der den KMS-Schlüssel angegeben hat. Die Parameter enthalten den ARN des vom Kunden verwalteten Schlüssels.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:sample-user",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/sample-user",
        "accountId": "111122223333",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
```

```
                "principalId": "AKIAIOSFODNN7EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-06-04T21:37:11Z"
            }
        },
        "invokedBy": "qldb.amazonaws.com"
    },
    "eventTime": "2021-06-04T21:40:00Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "qldb.amazonaws.com",
    "userAgent": "qldb.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "a30586af-c783-4d25-8fda-33152c816c36",
    "eventID": "7a9caf07-2b27-44ab-afe4-b259533ebb88",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

Verschlüsselung während der Übertragung in Amazon QLDB

Amazon QLDB akzeptiert nur sichere Verbindungen, die das HTTPS-Protokoll verwenden, das den Netzwerkverkehr mithilfe von Secure Sockets Layer (SSL)/Transport Layer Security (TLS) schützt. Die Verschlüsselung während der Übertragung bietet eine zusätzliche Datenschutzebene, indem Ihre Daten verschlüsselt werden, während sie zu und von QLDB übertragen werden. Organisationsrichtlinien, Branchen- oder behördliche Vorschriften sowie Compliance-Anforderungen erfordern häufig die Verwendung von Verschlüsselung bei der Übertragung, um die Datensicherheit Ihrer Anwendungen zu erhöhen, wenn sie Daten über das Netzwerk übertragen.

QLDB bietet auch FIPS-Endpunkte in ausgewählten Regionen an. Im Gegensatz zu Standard- AWS -Endpunkten verwenden FIPS-Endpunkte eine TLS-Softwarebibliothek, die den Federal Information Processing Standard (FIPS) 140-2 erfüllt. Diese Endpunkte können von Unternehmen erfordert werden, die mit der US-Regierung interagieren. Weitere Informationen finden Sie unter FIPS-Endpunkte im Allgemeine AWS-Referenz. Eine vollständige Liste der Regionen und Endpunkte, die für QLDB verfügbar sind, finden Sie unter Amazon-QLDB-Endpunkte und -Kontingente.

Identity and Access Management für Amazon QLDB

AWS Identity and Access Management (IAM) ist ein AWS-Service , mit dem ein Administrator den Zugriff auf - AWS Ressourcen sicher steuern kann. IAM-Administratoren steuern, wer für die Nutzung von QLDB-Ressourcen authentifiziert (angemeldet) und autorisiert (im Besitz von Berechtigungen) werden kann. IAM ist ein AWS-Service , den Sie ohne zusätzliche Kosten verwenden können.

Themen

- Fehlerbehebung für Amazon-QLDB-Identität und -Zugriff

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, unterscheidet sich je nach Ihrer Arbeit in QLDB.

Service-Benutzer – Wenn Sie den QLDB-Service zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen bereit, die Sie benötigen. Wenn Sie zur Ausführung von Aufgaben weitere QLDB-Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Fuktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Wenn Sie nicht auf ein Feature in QLDB zugreifen können, finden Sie weitere Informationen unter Fehlerbehebung für Amazon-QLDB-Identität und -Zugriff.

Service-Administrator – Wenn Sie in Ihrem Unternehmen für QLDB-Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollständigen Zugriff auf QLDB. Ihre Aufgabe besteht darin, zu bestimmen, auf welche QLDB-Funktionen und -Ressourcen Ihre Service-Benutzer zugreifen sollen. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen dazu, wie Ihr Unternehmen IAM mit QLDB verwenden kann, finden Sie unter Funktionsweise von Amazon QLDB mit IAM.

IAM-Administrator – Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf QLDB verfassen können. Beispiele für identitätsbasierte QLDB-Richtlinien, die Sie in IAM verwenden können, finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

Authentifizierung mit Identitäten

Die Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten bei anmelden. Sie müssen als Root-Benutzer des AWS-Kontos, als IAM-Benutzer oder durch Übernahme einer IAM-Rolle authentifiziert (bei angemeldet AWS) sein.

Sie können sich bei AWS als Verbundidentität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt werden. AWS IAM Identity Center (IAM Identity Center)-Benutzer, die Single-Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für Verbundidentitäten. Wenn Sie

sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie AWS über einen Verbund auf zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, um welchen Benutzertyp es sich handelt, können Sie sich bei der AWS Management Console oder im - AWS Zugriffsportal anmelden. Weitere Informationen zur Anmeldung bei AWSfinden Sie unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung - Benutzerhandbuch.

Wenn Sie AWS programmgesteuert auf zugreifen, AWS stellt ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (Command Line Interface, CLI) bereit, um Ihre Anforderungen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine - AWS Tools verwenden, müssen Sie Anforderungen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode zum eigenständigen Signieren von Anforderungen finden Sie unter Signieren von AWS API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. empfiehlt beispielsweise, AWS die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center - Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie ein erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services und Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet und Sie melden sich mit der E-Mail-Adresse und dem Passwort an, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Verbundidentität

Fordern Sie als bewährte Methode menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, auf, den Verbund mit einem Identitätsanbieter zu verwenden, um AWS-Services mithilfe temporärer Anmeldeinformationen auf zuzugreifen.

Eine Verbundidentität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, ein Web-Identitätsanbieter, die AWS Directory Service, das Identity-Center-Verzeichnis oder jeder Benutzer, der mit AWS-Services Anmeldeinformationen auf zugreift, die über eine Identitätsquelle bereitgestellt werden. Wenn Verbundidentitäten auf zugreifen AWS-Konten, übernehmen sie Rollen und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Centerzu verwenden. Sie können Benutzer und Gruppen in IAM Identity Center erstellen oder eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und synchronisieren, um sie für alle Ihre AWS-Konten und Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center -Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der übernehmen, AWS Management Console indem Sie die Rollen

wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI - oder AWS -API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können AWS-ServicesSie jedoch eine Richtlinie direkt an eine Ressource anfügen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff – Einige AWS-Services verwenden Funktionen in anderen AWS-Services. Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon EC2 aus oder speichert Objekte in Amazon S3. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward Access Sessions (FAS) – Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen in auszuführen AWS, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service auslösen. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anforderung AWS-Service , Anfragen an nachgelagerte Services zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Service eine Anfrage erhält, die Interaktionen mit anderen AWS-

Services oder -Ressourcen erfordert. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

- Servicerolle: Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
- Serviceverknüpfte Rolle – Eine serviceverknüpfte Rolle ist eine Art von Servicerolle, die mit einem verknüpft ist AWS-Service. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem angezeigt AWS-Konto und gehören dem Service. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

- Anwendungen, die auf Amazon EC2 ausgeführt werden – Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und - AWS CLI oder AWS -API-Anforderungen stellen. Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Um einer EC2-Instance eine - AWS Rolle zuzuweisen und sie für alle ihre Anwendungen verfügbar zu machen, erstellen Sie ein Instance-Profil, das der Instance zugeordnet ist. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie steuern den Zugriff in , AWS indem Sie Richtlinien erstellen und sie an AWS Identitäten oder Ressourcen anfügen. Eine Richtlinie ist ein Objekt in , AWS das, wenn es einer Identität oder Ressource zugeordnet wird, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anforderung stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können AWS JSON-Richtlinien verwenden, um anzugeben, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen über die AWS Management Console, die AWS CLIoder die AWS -API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem anfügen können AWS-Konto. Verwaltete Richtlinien umfassen - AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen

in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder umfassen AWS-Services.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Services AWS WAF, die ACLs unterstützen. Weitere Informationen zu ACLs finden Sie unter Zugriffssteuerungsliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger häufig verwendete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Service-Kontrollrichtlinien (SCPs)– SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in angeben AWS Organizations. AWS Organizations ist ein Service zum Gruppieren und zentralen Verwalten mehrerer AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Die SCP beschränkt Berechtigungen für Entitäten in Mitgliedskonten, einschließlich jeder Root-Benutzer des AWS-Kontos. Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations -Benutzerhandbuch.

- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Wie AWS bestimmt, ob eine Anforderung zugelassen werden soll, wenn mehrere Richtlinientypen beteiligt sind, erfahren Sie unter Logik zur Richtlinienbewertung im IAM-Benutzerhandbuch.

Funktionsweise von Amazon QLDB mit IAM

Bevor Sie IAM verwenden, um den Zugriff auf QLDB zu verwalten, erfahren Sie, welche IAM-Funktionen Sie mit QLDB verwenden können.

IAM-Funktionen, die Sie mit Amazon QLDB verwenden können

IAM-Feature	QLDB-Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Bedingungsschlüssel für die Richtlinie	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Ja
Temporäre Anmeldeinformationen	Ja

IAM-Feature	QLDB-Unterstützung
Hauptberechtigungen	Nein
Servicerollen	Ja
Service-verknüpfte Rollen	Nein

Einen Überblick über das AWS-Services Zusammenwirken von QLDB und anderen mit den meisten IAM-Funktionen finden Sie unter , AWS-Services die mit IAM funktionieren im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien für QLDB

Unterstützt Richtlinien auf Identitätsbasis.	Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für QLDB

Beispiele für identitätsbasierte QLDB-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

Ressourcenbasierte Richtlinien in QLDB

Unterstützt ressourcenbasierte Richtlinien	Nein

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder umfassen AWS-Services.

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource in unterschiedlichen befinden AWS-Konten, muss ein IAM-Administrator im vertrauenswürdigen Konto auch der Prinzipal-Entität (Benutzer oder Rolle) die Berechtigung für den Zugriff auf die Ressource erteilen. Sie erteilen Berechtigungen, indem Sie der juristischen Stelle eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie unter Wie sich IAM-Rollen von ressourcenbasierten Richtlinien unterscheiden im IAM-Benutzerhandbuch.

Richtlinienaktionen für QLDB

Unterstützt Richtlinienaktionen	Ja

Administratoren können AWS JSON-Richtlinien verwenden, um anzugeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben in der Regel denselben Namen wie die zugehörige AWS API-Operation. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keinen passenden API-Vorgang gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der QLDB-Aktionen finden Sie unter Von Amazon QLDB definierte Aktionen in der Service-Autorisierungs-Referenz.

Richtlinienaktionen in QLDB verwenden das folgende Präfix vor der Aktion:

```
qldb
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "qldb:action1",
      "qldb:action2"
]
```

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Describe` beginnen, einschließlich der folgenden Aktion:

```
"Action": "qldb:Describe*"
```

Um mit der QLDB-Transaktionsdaten-API (QLDB-Sitzung) zu interagieren, indem Sie PartiQL-Anweisungen in einem Ledger ausführen, müssen Sie der `SendCommand` Aktion wie folgt die Berechtigung erteilen.

```
"Action": "qldb:SendCommand"
```

Für Ledger im `STANDARD` Berechtigungsmodus finden Sie im Referenz zu PartiQL-Berechtigungen weitere erforderliche Berechtigungen für jeden PartiQL-Befehl.

Beispiele für identitätsbasierte QLDB-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

Richtlinienressourcen für QLDB

Unterstützt Richtlinienressourcen	Ja

Administratoren können AWS JSON-Richtlinien verwenden, um anzugeben, wer Zugriff auf was hat. Das bedeutet die Festlegung, welcher Prinzipal Aktionen für welche Ressourcen unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Liste der QLDB-Ressourcentypen und ihrer ARNs finden Sie unter Von Amazon QLDB definierte Ressourcen in der Service-Autorisierungs-Referenz. Informationen zu den Aktionen, mit denen Sie den ARN einzelner Ressourcen angeben können, finden Sie unter Von Amazon QLDB definierte Aktionen.

In QLDB sind die primären Ressourcen Ledger . QLDB unterstützt auch zusätzliche Ressourcentypen: Tabellen und Streams . Sie können jedoch Tabellen und Streams nur im Kontext eines vorhandenen Ledgers erstellen.

Eine QLDB-Tabelle ist eine materialisierte Ansicht einer ungeordneten Sammlung von Dokumentrevisionen aus dem Journal des Ledgers. Im `STANDARD` Berechtigungsmodus eines Ledgers müssen Sie IAM-Richtlinien erstellen, die Berechtigungen zum Ausführen von PartiQL-Anweisungen für diese Tabellenressource gewähren. Mit Berechtigungen für eine Tabellenressource können Sie Anweisungen ausführen, die auf den aktuellen Status der Tabelle zugreifen. Sie können den Revisionsverlauf der Tabelle auch mithilfe der integrierten `history()` Funktion abfragen. Weitere Informationen hierzu finden Sie unter Erste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

> **Note**
>
> Die `CREATE TABLE` Anweisung erstellt eine Tabelle mit einer eindeutigen ID und dem angegebenen Tabellennamen. Der angegebene Tabellenname muss unter allen aktiven

Tabellen eindeutig sein. Mit QLDB können Sie jedoch Tabellen deaktivieren, sodass es mehrere inaktive Tabellen geben kann, die denselben Tabellennamen haben. Daher beziehen sich Tabellenressourcen-ARNs auf die vom System zugewiesene eindeutige ID und nicht auf den benutzerdefinierten Tabellennamen.

Jeder Ledger bietet auch eine systemdefinierte Katalogressource, die Sie abfragen können, um alle Tabellen und Indizes in einem Ledger aufzulisten. Weitere Informationen zum QLDB-Datenobjektmodell finden Sie unter Kernkonzepte und Terminologie in Amazon QLDB.

Diesen Ressourcen sind eindeutige ARNs zugeordnet, wie in der folgenden Tabelle gezeigt.

Ressourcentyp	ARN
ledger	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}`
table	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/table/${TableId}`
catalog	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/information_schema/user_tables`
stream	`arn:${Partition}:qldb:${Region}:${Account}:stream/${LedgerName}/${StreamId}`

Um beispielsweise die `myExampleLedger` Ressource in Ihrer Anweisung anzugeben, verwenden Sie den folgenden ARN.

```
"Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
```

Um mehrere Ressourcen in einer einzigen Anweisung anzugeben, trennen Sie die ARNs durch Kommata voneinander.

```
"Resource": [
```

```
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger1",
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger2"
]
```

Beispiele für identitätsbasierte QLDB-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

Richtlinienbedingungsschlüssel für QLDB

Unterstützt servicespezifische Richtlinienbedingungsschlüssel	Ja

Administratoren können AWS JSON-Richtlinien verwenden, um anzugeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen `OR` Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und servicespezifische Bedingungsschlüssel. Informationen zum Anzeigen aller AWS globalen Bedingungsschlüssel finden Sie unter AWS Globale Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

Eine Liste der QLDB-Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für Amazon QLDB in der Service-Autorisierungs-Referenz. Informationen dazu, mit welchen Aktionen und Ressourcen

Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Von Amazon QLDB definierte Aktionen.

Die `PartiQLDropTable` Aktionen `PartiQLDropIndex` und unterstützen den `qldb:Purge` Bedingungsschlüssel . Dieser Bedingungsschlüssel filtert den Zugriff nach dem Wert von `purge`, der in einer PartiQL-`DROP`Anweisung angegeben ist. QLDB unterstützt derzeit jedoch nur `purge = true` für -`DROP INDEX`Anweisungen und `purge = false` für -`DROP TABLE`Anweisungen. Andere QLDB-Aktionen unterstützen einige globale Bedingungsschlüssel.

Beispiele für identitätsbasierte QLDB-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

Zugriffssteuerungslisten (ACLs) in QLDB

Unterstützt ACLs	Nein

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Attributbasierte Zugriffskontrolle (ABAC) mit QLDB

Unterstützt ABAC (Tags in Richtlinien)	Ja

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. In werden AWSdiese Attribute als Tags bezeichnet. Sie können Tags an IAM-Entitäten (Benutzer oder Rollen) und an viele AWS Ressourcen anfügen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt von ABAC. Anschließend entwerfen Sie ABAC-Richtlinien, um Operationen zuzulassen, wenn das Tag des Prinzipals mit dem Tag der Ressource übereinstimmt, auf die sie zugreifen möchten.

ABAC ist in Umgebungen hilfreich, die schnell wachsen, und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Was ist ABAC? im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Weitere Informationen zum Markieren von QLDB-Ressourcen finden Sie unter Markieren von Amazon-QLDB Ressourcen.

Ein Beispiel für eine identitätsbasierte Richtlinie zur Einschränkung des Zugriffs auf eine Ressource auf der Grundlage der Markierungen dieser Ressource finden Sie unter Aktualisieren von QLDB-Ledgern basierend auf Tags.

Verwenden temporärer Anmeldeinformationen mit QLDB

Unterstützt temporäre Anmeldeinformationen	Ja

Einige funktionieren AWS-Services nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, einschließlich der , die mit temporären Anmeldeinformationen AWS-Services funktionieren, finden Sie unter AWS-Services , die mit IAM funktionieren im IAM-Benutzerhandbuch.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich AWS Management Console mit einer anderen Methode als einem Benutzernamen und einem Passwort bei der anmelden. Wenn Sie beispielsweise AWS über den SSO-Link (Single Sign-On) Ihres Unternehmens auf zugreifen, erstellt dieser Prozess automatisch temporäre Anmeldeinformationen. Sie erstellen auch automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden und dann die Rollen wechseln. Weitere Informationen zum Wechseln von Rollen finden Sie unter Wechseln zu einer Rolle (Konsole) im IAM-Benutzerhandbuch.

Sie können temporäre Anmeldeinformationen manuell mit der AWS CLI oder der AWS API erstellen. Sie können diese temporären Anmeldeinformationen dann verwenden, um auf zuzugreifen AWS. AWS empfohlen, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen in IAM.

Serviceübergreifende Prinzipal-Berechtigungen für QLDB

Unterstützt Forward Access Sessions (FAS)	Nein

Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen in auszuführen AWS, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service auslösen. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anforderung AWS-Service , Anfragen an nachgelagerte Services zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Service eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder -Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für QLDB

Unterstützt Servicerollen	Ja

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

> **Warning**
>
> Das Ändern der Berechtigungen für eine Servicerolle könnte die QLDB-Funktionalität beeinträchtigen. Bearbeiten Sie Servicerollen nur, wenn QLDB dazu Anleitungen gibt.

QLDB unterstützt Servicerollen für die API`StreamJournalToKinesis`-Operationen `ExportJournalToS3` und , wie im folgenden Abschnitt beschrieben.

Auswählen einer IAM-Rolle in QLDB

Wenn Sie Journalblöcke in QLDB exportieren oder streamen, müssen Sie eine Rolle auswählen, damit QLDB Objekte in Ihrem Namen in das angegebene Ziel schreiben kann. Wenn Sie zuvor eine

Servicerolle erstellt haben, stellt Ihnen QLDB eine Liste der Rollen zur Auswahl bereit. Es ist wichtig, eine Rolle auszuwählen, die den Zugriff auf den angegebenen Amazon S3-Bucket für einen Export oder auf die von Ihnen angegebene Amazon Kinesis-Data-Streams-Ressource für einen Stream ermöglicht. Weitere Informationen finden Sie unter Journalexportberechtigungen in QLDB oder Stream-Berechtigungen in QLDB.

> **Note**
>
> Um beim Anfordern eines Journalexports oder -streams eine Rolle an QLDB zu übergeben, müssen Sie über Berechtigungen zum Ausführen der `iam:PassRole` Aktion für die IAM-Rollenressource verfügen. Dies gilt zusätzlich zu den Berechtigungen, die `qldb:ExportJournalToS3` für die QLDB-Ledger-Ressource oder `qldb:StreamJournalToKinesis` für die QLDB-Stream-Subressource ausgeführt werden können.

Serviceverknüpfte Rollen für QLDB

Unterstützt serviceverknüpfte Rollen	Nein

Eine serviceverknüpfte Rolle ist eine Art von Servicerolle, die mit einem verknüpft ist AWS-Service. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem angezeigt AWS-Konto und gehören dem Service. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Weitere Informationen zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter AWS-Services , die mit IAM funktionieren. Suchen Sie in der Tabelle nach einem Service mit einem `Yes` in der Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Erste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB

Verwenden Sie diesen Abschnitt, um mit dem Standard-Berechtigungsmodus in Amazon QLDB zu beginnen. Dieser Abschnitt enthält eine Referenztabelle, die Ihnen beim Schreiben einer identitätsbasierten Richtlinie in AWS Identity and Access Management (IAM) für PartiQL-Aktionen und Tabellenressourcen in QLDB hilft. Sie enthält auch ein step-by-step Tutorial zum Erstellen von

Berechtigungsrichtlinien in IAM sowie Anweisungen zum Suchen eines Tabellen-ARN und zum Erstellen von Tabellen-Tags in QLDB.

Themen

Der `STANDARD` Berechtigungsmodus

QLDB unterstützt jetzt einen `STANDARD` Berechtigungsmodus für Ledger-Ressourcen. Der Standardberechtigungsmodus ermöglicht die Zugriffskontrolle mit feinerer Granularität für Ledger, Tabellen und PartiQL-Befehle. Standardmäßig verweigert dieser Modus alle Anforderungen zum Ausführen von PartiQL-Befehlen für alle Tabellen in einem Ledger.

> **Note**
>
> Zuvor war der einzige verfügbare Berechtigungsmodus für einen Ledger `ALLOW_ALL`. Der `-ALLOW_ALL`Modus ermöglicht die Zugriffskontrolle mit Granularität auf API-Ebene für Ledger und wird weiterhin – aber nicht empfohlen – für QLDB-Ledger unterstützt. In diesem Modus können Benutzer mit der `SendCommand` API-Berechtigung alle PartiQL-Befehle für alle Tabellen im Ledger ausführen, die in der Berechtigungsrichtlinie angegeben sind (und somit alle PartiQL-Befehle zulassen).
> Sie können den Berechtigungsmodus vorhandener Ledger von `ALLOW_ALL` in ändern`STANDARD`. Weitere Informationen finden Sie unter Migration in den Standardberechtigungsmodus.

Um Befehle im Standardmodus zuzulassen, müssen Sie in IAM eine Berechtigungsrichtlinie für bestimmte Tabellenressourcen und PartiQL-Aktionen erstellen. Dies gilt zusätzlich zur `SendCommand` API-Berechtigung für das Ledger. Um Richtlinien in diesem Modus zu erleichtern, führte QLDB eine Reihe von IAM-Aktionen für PartiQL-Befehle und Amazon-Ressourcennamen (ARNs) für QLDB-Tabellen ein. Weitere Informationen zum QLDB-Datenobjektmodell finden Sie unter Kernkonzepte und Terminologie in Amazon QLDB.

Referenz zu PartiQL-Berechtigungen

In der folgenden Tabelle sind die einzelnen QLDB-PartiQL-Befehle, die entsprechenden IAM-Aktionen, für die Sie Berechtigungen zum Ausführen des Befehls erteilen müssen, und die AWS Ressourcen, für die Sie die Berechtigungen erteilen können, aufgeführt. Die Aktionen geben Sie im Feld `Action` und den Wert für die Ressource im Feld `Resource` der Richtlinie an.

Important

- IAM-Richtlinien, die Berechtigungen für diese PartiQL-Befehle gewähren, gelten nur für Ihr Ledger, wenn dem Ledger der `STANDARD` Berechtigungsmodus zugewiesen ist. Solche Richtlinien gelten nicht für Ledger im `ALLOW_ALL` Berechtigungsmodus.

  Informationen zum Angeben des Berechtigungsmodus beim Erstellen oder Aktualisieren eines Ledgers finden Sie unter Grundfunktionen für Amazon QLDB-Ledgersoder Schritt 1: Erstellen eines neuen Ledgers unter Erste Schritte mit der Konsole.
- Um PartiQL-Befehle auf einem Ledger auszuführen, müssen Sie auch die Berechtigung für die `SendCommand` API-Aktion für die Ledger-Ressource erteilen. Dies gilt zusätzlich zu den PartiQL-Aktionen und Tabellenressourcen, die in der folgenden Tabelle aufgeführt sind. Weitere Informationen finden Sie unter Ausführen von Datentransaktionen.

Amazon-QLDB-PartiQL-Befehle und erforderliche Berechtigungen

Befehl	Erforderliche Berechtigungen (IAM-Aktionen)	Ressourcen	Abhängige Aktionen
CREATE TABLE	`qldb:PartiQLCreateTable`	`arn:aws:qldb:` *region*`:`*account-id*`:ledger/`*ledger-name*`/table/*`	`qldb:TagResource` (zum Markieren bei der

Befehl	Erforderliche Berechtigungen (IAM-Aktionen)	Ressourcen	Abhängige Aktionen
			Erstellung)
DROP TABLE	qldb:PartiQLDropTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
UNDROP TABLE	qldb:PartiQLUndropTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
CREATE INDEX	qldb:PartiQLCreateIndex	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
DROP INDEX	qldb:PartiQLDropIndex	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
DELETE FROM-REMOVE (für ganze Dokumente)	qldb:PartiQLDelete	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	qldb:PartiQLSelect
INSERT	qldb:PartiQLInsert	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	

Befehl	Erforderliche Berechtigungen (IAM-Aktionen)	Ressourcen	Abhängige Aktionen
UPDATE VON (INSERT, REMOVE oder SET)	qldb:PartiQLUpdate	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	qldb:PartiQLSelect
REVISION REDIGIERIN (gespeicherte Prozedur)	qldb:PartiQLRedact	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
SELECT FROM table_name	qldb:PartiQLSelect	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
SELECT FROM information_schema.user_tables	qldb:PartiQLSelect	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /information_schema/user_tables	

Befehl	Erforderliche Berechtigungen (IAM-Aktionen)	Ressourcen	Abhängige Aktionen
SELECT FROM history(table_name)	qldb:PartiQLHistoryFunction	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	

Beispiele für IAM-Richtliniendokumente, die Berechtigungen für diese PartiQL-Befehle gewähren, finden Sie unter Schnellstart-Tutorial: Erstellen von Berechtigungsrichtlinien oder unter Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

Suchen einer Tabellen-ID und eines ARN

Sie können eine Tabellen-ID finden, indem Sie die verwenden AWS Management Console oder die Tabelle information_schema.user_tables abfragen. Um Tabellendetails in der Konsole anzuzeigen oder diese Systemkatalogtabelle abzufragen, benötigen Sie die -SELECTBerechtigung für die Systemkatalogressource. Um beispielsweise die Tabellen-ID der `Vehicle` Tabelle zu finden, können Sie die folgende Anweisung ausführen.

```
SELECT * FROM information_schema.user_tables
WHERE name = 'Vehicle'
```

Diese Abfrage gibt Ergebnisse in einem Format zurück, das dem folgenden Beispiel ähnelt.

```
{
    tableId: "Au1EiThbt8s0z9wM26REZN",
    name: "Vehicle",
    indexes: [
        { indexId: "Djg2nt0yIs2GY0T29Kud1z", expr: "[VIN]", status: "ONLINE" },
        { indexId: "4tPW3fUhaVhDinRgKRLhGU", expr: "[LicensePlateNumber]", status:
 "BUILDING" }
    ],
    status: "ACTIVE"
```

```
}
```

Um Berechtigungen zum Ausführen von PartiQL-Anweisungen für eine Tabelle zu erteilen, geben Sie eine Tabellenressource im folgenden ARN-Format an.

```
arn:aws:qldb:${region}:${account-id}:ledger/${ledger-name}/table/${table-id}
```

Im Folgenden finden Sie ein Beispiel für einen Tabellen-ARN für die Tabellen-ID `Au1EiThbt8s0z9wM26REZN`.

```
arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/Au1EiThbt8s0z9wM26REZN
```

Verwenden der Konsole

Sie können auch die QLDB-Konsole verwenden, um einen Tabellen-ARN zu finden.

So finden Sie den ARN einer Tabelle (Konsole)

1. Melden Sie sich bei der an AWS Management Consoleund öffnen Sie die Amazon-QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich Ledgers aus.
3. Wählen Sie in der Liste der Ledger den Ledger-Namen aus, dessen Tabellen-ARN Sie suchen möchten.
4. Suchen Sie auf der Ledger-Detailseite auf der Registerkarte Tabellen den Tabellennamen, dessen ARN Sie finden möchten. Um den ARN zu kopieren, wählen Sie das Kopiersymbol () daneben aus.

Markieren von Tabellen

Sie können Ihre Tabellenressourcen markieren. Um Tags für vorhandene Tabellen zu verwalten, verwenden Sie die AWS Management Console oder die API-Operationen `TagResource`, `UntagResource`und `ListTagsForResource`. Weitere Informationen finden Sie unter Markieren von Amazon-QLDB Ressourcen.

> **Note**
>
> Tabellenressourcen erben nicht die Tags ihrer Root-Ledger-Ressource.

> Das Markieren von Tabellen bei der Erstellung wird derzeit nur für Ledger im `STANDARD` Berechtigungsmodus unterstützt.

Sie können Tabellen-Tags auch definieren, während Sie die Tabelle erstellen, indem Sie die QLDB-Konsole verwenden oder sie in einer `CREATE TABLE` PartiQL-Anweisung angeben. Im folgenden Beispiel wird eine Tabelle mit dem Namen `Vehicle` mit dem Tag erstellt`environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Das Markieren von Tabellen bei der Erstellung erfordert Zugriff auf die `qldb:TagResource` Aktionen `qldb:PartiQLCreateTable` und .

Indem Sie Ressourcen zum Erstellungszeitpunkt markieren, müssen Sie anschließend keine benutzerdefinierten Markierungs-Skripts ausführen. Nachdem eine Tabelle markiert wurde, können Sie den Zugriff auf die Tabelle anhand dieser Tags steuern. Sie können beispielsweise nur für Tabellen mit einem bestimmten Tag Vollzugriff gewähren. Ein Beispiel für eine JSON-Richtlinie finden Sie unter Vollzugriff auf alle Aktionen basierend auf Tabellen-Tags.

Verwenden der Konsole

Sie können auch die QLDB-Konsole verwenden, um Tabellen-Tags zu definieren, während Sie die Tabelle erstellen.

So markieren Sie eine Tabelle bei der Erstellung (Konsole)

1. Melden Sie sich bei der an AWS Management Consoleund öffnen Sie die Amazon-QLDB-Konsole unter https://console.aws.amazon.com/qldb.
2. Wählen Sie im Navigationsbereich Ledgers aus.
3. Wählen Sie in der Liste der Ledger den Ledger-Namen aus, in dem Sie die Tabelle erstellen möchten.
4. Wählen Sie auf der Ledger-Detailseite auf der Registerkarte Tabellen die Option Tabelle erstellen aus.
5. Gehen Sie auf der Seite Tabelle erstellen wie folgt vor:

   - Tabellenname – Geben Sie einen Tabellennamen ein.

- Tags – Fügen Sie der Tabelle Metadaten hinzu, indem Sie Tags als Schlüssel-Wert-Paare anfügen. Sie können Ihrer Tabelle Tags hinzufügen, um sie zu organisieren und zu identifizieren.

  Wählen Sie Add tag (Tag hinzufügen) aus und geben Sie dann beliebige Schlüssel-Wert-Paare ein.

6. Wenn Sie die gewünschten Einstellungen vorgenommen haben, wählen Sie Create table (Tabelle erstellen).

Schnellstart-Tutorial: Erstellen von Berechtigungsrichtlinien

Dieses Tutorial führt Sie durch die Schritte zum Erstellen von Berechtigungsrichtlinien in IAM für einen Amazon-QLDB-Ledger im `STANDARD` Berechtigungsmodus. Anschließend können Sie Ihren Benutzern, Gruppen oder Rollen die Berechtigungen zuweisen.

Weitere Beispiele für IAM-Richtliniendokumente, die PartiQL-Befehlen und Tabellenressourcen Berechtigungen gewähren, finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

Themen

Voraussetzungen

Bevor Sie beginnen, stellen Sie sicher, dass die folgende Voraussetzung erfüllt ist:

1. Folgen Sie den AWS Einrichtungsanweisungen unter Zugreifen auf Amazon QLDB, falls Sie dies noch nicht getan haben. Zu diesen Schritten gehören die Registrierung für AWS und die Erstellung eines Administratorbenutzers.
2. Erstellen Sie ein neues Ledger und wählen Sie den `STANDARD` Berechtigungsmodus für das Ledger aus. Weitere Informationen finden Sie unter Schritt 1: Erstellen eines neuen Ledgers in Erste Schritte mit der Konsole oder Grundfunktionen für Amazon QLDB-Ledgers.

Erstellen einer schreibgeschützten Richtlinie

Gehen Sie wie folgt vor, um mit dem JSON-Richtlinieneditor eine schreibgeschützte Richtlinie für alle Tabellen in einem Ledger im Standardberechtigungsmodus zu erstellen:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste auf der linken Seite auf Policies (Richtlinien).

   Wenn Sie zum ersten Mal Policies (Richtlinien) auswählen, erscheint die Seite Welcome to Managed Policies (Willkommen bei verwalteten Richtlinien). Wählen Sie Get Started.
3. Wählen Sie oben auf der Seite Create policy (Richtlinie erstellen) aus.
4. Wählen Sie den Tab JSON.
5. Kopieren Sie das folgende JSON-Richtliniendokument und fügen Sie es ein. Diese Beispielrichtlinie gewährt schreibgeschützten Zugriff auf alle Tabellen in einem Ledger.

   Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012* und *myExampleLedger* im Beispiel durch Ihre eigenen Informationen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "QLDBSendCommandPermission",
      "Effect": "Allow",
      "Action": "qldb:SendCommand",
      "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
    },
    {
      "Sid": "QLDBPartiQLReadOnlyPermissions",
      "Effect": "Allow",
      "Action": [
        "qldb:PartiQLSelect",
        "qldb:PartiQLHistoryFunction"
      ],
      "Resource": [
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
      ]
    }
```

```
  ]
}
```

6. Wählen Sie Richtlinie prüfen.

> **Note**
>
> Sie können jederzeit zwischen den Registerkarten Visual editor (Visueller Editor) und JSON wechseln. Wenn Sie jedoch Änderungen vornehmen oder auf der Registerkarte Visueller Editor die Option Richtlinie überprüfen auswählen, strukturiert IAM möglicherweise Ihre Richtlinie neu, um sie für den visuellen Editor zu optimieren. Weitere Informationen finden Sie unter Richtlinienrestrukturierung im IAM-Benutzerhandbuch.

7. Geben Sie auf der Seite Review policy (Richtlinie überprüfen) unter Name einen Namen und unter Description (Beschreibung) eine optionale Beschreibung für die Richtlinie ein, die Sie erstellen. Überprüfen Sie unter Summary die Richtlinienzusammenfassung, um die Berechtigungen einzusehen, die von Ihrer Richtlinie gewährt werden. Wählen Sie dann Create policy aus, um Ihre Eingaben zu speichern.

Erstellen einer Vollzugriffsrichtlinie

Gehen Sie wie folgt vor, um eine Vollzugriffsrichtlinie für alle Tabellen in einem QLDB-Ledger im Standardberechtigungsmodus zu erstellen:

- Wiederholen Sie die vorherigen Schritte mit dem folgenden Richtliniendokument. Diese Beispielrichtlinie gewährt Zugriff auf alle PartiQL-Befehle für alle Tabellen in einem Ledger, indem Platzhalter (*) verwendet werden, um alle PartiQL-Aktionen und alle Ressourcen unter einem Ledger abzudecken.

> **Warning**
>
> Dies ist ein Beispiel für die Verwendung eines Platzhalterzeichens (*), um alle PartiQL-Aktionen zuzulassen, einschließlich administrativer und Lese-/Schreibvorgänge für alle Tabellen in einem QLDB-Ledger. Stattdessen empfiehlt es sich, jede Aktion explizit anzugeben, die gewährt werden soll, und nur was dieser Benutzer, diese Rolle oder diese Gruppe benötigt.

Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012* und *myExampleLedger* im Beispiel durch Ihre eigenen Informationen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "QLDBSendCommandPermission",
      "Effect": "Allow",
      "Action": "qldb:SendCommand",
      "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
    },
    {
      "Sid": "QLDBPartiQLFullPermissions",
      "Effect": "Allow",
      "Action": [
        "qldb:PartiQL*"
      ],
      "Resource": [
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
      ]
    }
  ]
}
```

Erstellen einer schreibgeschützten Richtlinie für eine bestimmte Tabelle

Gehen Sie wie folgt vor, um eine schreibgeschützte Zugriffsrichtlinie für eine bestimmte Tabelle in einem QLDB-Ledger im Standardberechtigungsmodus zu erstellen:

1. Suchen Sie den ARN für die Tabelle mithilfe der AWS Management Console oder durch Abfragen der Systemkatalogtabelle `information_schema.user_tables`. Anweisungen finden Sie unter Suchen einer Tabellen-ID und eines ARN.
2. Verwenden Sie den Tabellen-ARN, um eine Richtlinie zu erstellen, die schreibgeschützten Zugriff auf die Tabelle erlaubt. Wiederholen Sie dazu die vorherigen Schritte mit dem folgenden Richtliniendokument.

Diese Beispielrichtlinie gewährt nur Lesezugriff auf die angegebene Tabelle . In diesem Beispiel lautet die Tabellen-ID `Au1EiThbt8s0z9wM26REZN`. Um diese Richtlinie zu verwenden, ersetzen Sie *`us-east-1`*, *`123456789012myExampleLedger`*, und *`Au1EiThbt8s0z9wM26REZN`* im Beispiel durch Ihre eigenen Informationen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "QLDBSendCommandPermission",
      "Effect": "Allow",
      "Action": "qldb:SendCommand",
      "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
    },
    {
      "Sid": "QLDBPartiQLReadOnlyPermissions",
      "Effect": "Allow",
      "Action": [
        "qldb:PartiQLSelect",
        "qldb:PartiQLHistoryFunction"
      ],
      "Resource": [
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
table/Au1EiThbt8s0z9wM26REZN"
      ]
    }
  ]
}
```

Zuweisen von Berechtigungen

Nachdem Sie eine QLDB-Berechtigungsrichtlinie erstellt haben, weisen Sie die Berechtigungen wie folgt zu.

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für Amazon QLDB

Benutzer und Rollen besitzen standardmäßig keine Berechtigungen zum Erstellen oder Ändern von QLDB-Ressourcen. Sie können auch keine Aufgaben mithilfe der AWS Management Console, AWS Command Line Interface (AWS CLI) oder AWS API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Einzelheiten zu Aktionen und Ressourcentypen, die von QLDB definiert werden, einschließlich des Formats der ARNs für die einzelnen Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon QLDB in der Service-Autorisierungs-Referenz.

Inhalt

    - Vollzugriff auf alle Aktionen
    - Vollzugriff auf alle Aktionen basierend auf Tabellen-Tags
    - Lese-/Schreibzugriff
    - Schreibgeschützter Zugriff
    - Schreibgeschützter Zugriff auf eine bestimmte Tabelle
    - Zugriff zum Erstellen von Tabellen zulassen
    - Erlauben des Zugriffs zum Erstellen von Tabellen basierend auf Anforderungs-Tags
- Exportieren eines Journals in einen Amazon S3-Bucket
- Streamen eines Journals zu Kinesis Data Streams
- Aktualisieren von QLDB-Ledgern basierend auf Tags

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand QLDB-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Erste Schritte mit AWS -verwalteten Richtlinien und Umstellung auf Berechtigungen mit den geringsten Berechtigungen – Um Ihren Benutzern und Workloads Berechtigungen zu erteilen, verwenden Sie die -AWS verwalteten Richtlinien, die Berechtigungen für viele häufige Anwendungsfälle gewähren. Sie sind in Ihrem verfügbar AWS-Konto. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die für Ihre Anwendungsfälle spezifisch sind. Weitere Informationen finden Sie unter AWS -verwaltete Richtlinien oder AWS -verwaltete Richtlinien für Auftrags-Funktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten:Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.

- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Service-Aktionen zu gewähren, wenn sie über eine bestimmte verwendet werden AWS-Service, z. B. AWS CloudFormation. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung zum IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich – Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Konfigurieren eines MFA-geschützten API-Zugriffs im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der QLDB-Konsole

Um auf die Amazon-QLDB-Konsole zugreifen zu können, müssen Sie über einen Mindestsatz von Berechtigungen verfügen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den QLDB-Ressourcen in Ihrem aufzulisten und anzuzeigen AWS-Konto. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Für Benutzer, die nur Aufrufe an die AWS CLI oder die AWS API durchführen, müssen Sie keine Mindestberechtigungen für die Konsole erteilen. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Um sicherzustellen, dass Benutzer und Rollen vollen Zugriff auf die QLDB-Konsole und alle ihre Funktionen haben, fügen Sie den Entitäten die folgende AWS verwaltete Richtlinie hinzu. Weitere Informationen finden Sie unter AWS Von verwaltete Richtlinien für Amazon QLDBund Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch.

```
AmazonQLDBConsoleFullAccess
```

Berechtigungen für den Abfrageverlauf

Zusätzlich zu QLDB-Berechtigungen erfordern einige Konsolenfunktionen Berechtigungen für den Database Query Metadata Service (Servicepräfix: `dbqms`). Dies ist ein interner Service, der Ihre letzten und gespeicherten Abfragen im Abfrage-Editor der Konsole für QLDB und andere verwaltet AWS-Services. Eine vollständige Liste der DBQMS-API-Aktionen finden Sie unter Database Query Metadata Service in der Service-Autorisierungs-Referenz.

Um Berechtigungen für den Abfrageverlauf zu gewähren, können Sie die von AWS verwaltete Richtlinie AmazonQLDBConsoleFullAccess verwenden. Diese Richtlinie verwendet einen Platzhalter (`dbqms:*`), um alle DBQMS-Aktionen für alle Ressourcen zuzulassen.

Oder Sie können eine benutzerdefinierte IAM-Richtlinie erstellen und die folgenden DBQMS-Aktionen einschließen. Der PartiQL-Abfrage-Editor in der QLDB-Konsole benötigt Berechtigungen, um diese Aktionen für Abfrageverlaufsfunktionen zu verwenden.

```
dbqms:CreateFavoriteQuery
dbqms:CreateQueryHistory
dbqms:DeleteFavoriteQueries
dbqms:DeleteQueryHistory
dbqms:DescribeFavoriteQueries
dbqms:DescribeQueryHistory
dbqms:UpdateFavoriteQuery
```

Konsolenberechtigungen mit vollem Zugriff ohne Abfrageverlauf

Um vollen Zugriff auf die QLDB-Konsole ohne Berechtigungen für den Abfrageverlauf zu gewähren, können Sie eine benutzerdefinierte IAM-Richtlinie erstellen, die alle DBQMS-Aktionen ausschließt. Das folgende Richtliniendokument erlaubt beispielsweise dieselben Berechtigungen, die von der AWS verwalteten Richtlinie AmazonQLDBConsoleFullAccess gewährt werden, mit Ausnahme von Aktionen, die mit dem Servicepräfix beginnen`dbqms`.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "qldb:CreateLedger",
        "qldb:UpdateLedger",
        "qldb:UpdateLedgerPermissionsMode",
        "qldb:DeleteLedger",
        "qldb:ListLedgers",
        "qldb:DescribeLedger",
        "qldb:ExportJournalToS3",
        "qldb:ListJournalS3Exports",
        "qldb:ListJournalS3ExportsForLedger",
        "qldb:DescribeJournalS3Export",
        "qldb:CancelJournalKinesisStream",
        "qldb:DescribeJournalKinesisStream",
        "qldb:ListJournalKinesisStreamsForLedger",
        "qldb:StreamJournalToKinesis",
        "qldb:GetBlock",
        "qldb:GetDigest",
        "qldb:GetRevision",
        "qldb:TagResource",
        "qldb:UntagResource",
        "qldb:ListTagsForResource",
        "qldb:SendCommand",
        "qldb:ExecuteStatement",
        "qldb:ShowCatalog",
        "qldb:InsertSampleData",
        "qldb:PartiQLCreateIndex",
        "qldb:PartiQLDropIndex",
        "qldb:PartiQLCreateTable",
        "qldb:PartiQLDropTable",
        "qldb:PartiQLUndropTable",
        "qldb:PartiQLDelete",
        "qldb:PartiQLInsert",
        "qldb:PartiQLUpdate",
        "qldb:PartiQLSelect",
        "qldb:PartiQLHistoryFunction"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": [
```

```
        "kinesis:ListStreams",
        "kinesis:DescribeStream"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "qldb.amazonaws.com"
        }
      }
    }
  ]
}
```

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie enthält Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der AWS CLI oder AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
```

```
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Ausführen von Datentransaktionen

Um mit der QLDB-Transaktionsdaten-API (QLDB-Sitzung) zu interagieren, indem Sie PartiQL-Anweisungen in einem Ledger ausführen, müssen Sie der `SendCommand` API-Aktion die Berechtigung erteilen. Das folgende JSON-Dokument ist ein Beispiel für eine Richtlinie, die nur die Berechtigung für die `SendCommand` API-Aktion im Ledger gewährt`myExampleLedger`.

Um diese Richtlinie zu verwenden, ersetzen Sie *`us-east-1`*, *`123456789012`* und *`myExampleLedger`* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        }
    ]
}
```

Wenn den `ALLOW_ALL` Berechtigungsmodus `myExampleLedger` verwendet, gewährt diese Richtlinie Berechtigungen zum Ausführen aller PartiQL-Befehle für jede Tabelle im Ledger.

Sie können auch eine von AWS verwaltete Richtlinie verwenden, um vollen Zugriff auf alle QLDB-Ressourcen zu gewähren. Weitere Informationen finden Sie unter AWS Von verwaltete Richtlinien für Amazon QLDB.

Standardberechtigungen für PartiQL-Aktionen und Tabellenressourcen

Für Ledger im `STANDARD` Berechtigungsmodus können Sie sich auf die folgenden IAM-Richtliniendokumente beziehen, um Beispiele für die Erteilung der entsprechenden PartiQL-Berechtigungen zu erhalten. Eine Liste der erforderlichen Berechtigungen für jeden PartiQL-Befehl finden Sie unter Referenz zu PartiQL-Berechtigungen.

Themen

- Vollzugriff auf alle Aktionen
- Vollzugriff auf alle Aktionen basierend auf Tabellen-Tags
- Lese-/Schreibzugriff
- Schreibgeschützter Zugriff
- Schreibgeschützter Zugriff auf eine bestimmte Tabelle
- Zugriff zum Erstellen von Tabellen zulassen
- Erlauben des Zugriffs zum Erstellen von Tabellen basierend auf Anforderungs-Tags

Vollzugriff auf alle Aktionen

Das folgende JSON-Richtliniendokument gewährt vollen Zugriff auf die Verwendung all erPartiQL-Befehle für all eTabellen in `myExampleLedger`. Diese Richtlinie hat den gleichen Effekt wie die Verwendung des `ALLOW_ALL` Berechtigungsmodus für das Ledger.

Um diese Richtlinie zu verwenden, ersetzen Sie *`us-east-1`*, *`123456789012`* und *`myExampleLedger`* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
```

```
        },
        {
            "Sid": "QLDBPartiQLFullPermissions",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLCreateIndex",
                "qldb:PartiQLDropIndex",
                "qldb:PartiQLCreateTable",
                "qldb:PartiQLDropTable",
                "qldb:PartiQLUndropTable",
                "qldb:PartiQLDelete",
                "qldb:PartiQLInsert",
                "qldb:PartiQLUpdate",
                "qldb:PartiQLRedact",
                "qldb:PartiQLSelect",
                "qldb:PartiQLHistoryFunction"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
            ]
        }
    ]
}
```

Vollzugriff auf alle Aktionen basierend auf Tabellen-Tags

Das folgende JSON-Richtliniendokument verwendet eine Bedingung, die auf Tabellenressourcen-Tags basiert, um vollen Zugriff auf die Verwendung all erPartiQL-Befehle für all eTabellen in zu gewährenmyExampleLedger. Berechtigungen werden nur erteilt, wenn das Tabellen-Tag den Wert `environment` hat`development`.

> **Warning**
>
> Dies ist ein Beispiel für die Verwendung eines Platzhalterzeichens (*), um alle PartiQL-Aktionen zuzulassen, einschließlich administrativer und Lese-/Schreibvorgänge für alle Tabellen in einem QLDB-Ledger. Stattdessen empfiehlt es sich, jede Aktion explizit anzugeben, die gewährt werden soll, und nur was dieser Benutzer, diese Rolle oder diese Gruppe benötigt.

Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012* und *myExampleLedger* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLFullPermissionsBasedOnTags",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQL*"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
            ],
            "Condition": {
                "StringEquals": { "aws:ResourceTag/environment": "development" }
            }
        }
    ]
}
```

Lese-/Schreibzugriff

Das folgende JSON-Richtliniendokument gewährt Berechtigungen zum Auswählen, Einfügen, Aktualisieren und Löschen von Daten für alle Tabellen in `myExampleLedger`. Diese Richtlinie gewährt keine Berechtigungen, um Daten zu redigieren oder das Schema zu ändern, z. B. das Erstellen und Löschen von Tabellen und Indizes.

> **Note**
>
> Eine -`UPDATE`Anweisung erfordert Berechtigungen für die `qldb:PartiQLSelect` Aktionen `qldb:PartiQLUpdate` und für die Tabelle, die geändert wird. Wenn Sie eine -`UPDATE`Anweisung ausführen, führt sie zusätzlich zum Aktualisierungsvorgang eine

Leseoperation aus. Die Erzwingung beider Aktionen stellt sicher, dass nur Benutzern, die den Inhalt einer Tabelle lesen dürfen, UPDATE Berechtigungen gewährt werden.
In ähnlicher Weise erfordert eine DELETE Anweisung Berechtigungen sowohl für die Aktionen als auch für die `qldb:PartiQLDelete` `qldb:PartiQLSelect` Aktionen .

Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012* und *myExampleLedger* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLReadWritePermissions",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLDelete",
                "qldb:PartiQLInsert",
                "qldb:PartiQLUpdate",
                "qldb:PartiQLSelect",
                "qldb:PartiQLHistoryFunction"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
            ]
        }
    ]
}
```

Schreibgeschützter Zugriff

Das folgende JSON-Richtliniendokument gewährt Leseberechtigungen für alle Tabellen in `myExampleLedger`. Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012* und *myExampleLedger* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLReadOnlyPermissions",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLSelect",
                "qldb:PartiQLHistoryFunction"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
            ]
        }
    ]
}
```

Schreibgeschützter Zugriff auf eine bestimmte Tabelle

Das folgende JSON-Richtliniendokument gewährt Leseberechtigungen für eine bestimmte Tabelle in `myExampleLedger`. In diesem Beispiel lautet die Tabellen-ID `Au1EiThbt8s0z9wM26REZN`.

Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012myExampleLedger*, und *Au1EiThbt8s0z9wM26REZN* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
```

```
            "Sid": "QLDBPartiQLReadOnlyPermissionsOnTable",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLSelect",
                "qldb:PartiQLHistoryFunction"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
table/Au1EiThbt8s0z9wM26REZN"
            ]
        }
    ]
}
```

Zugriff zum Erstellen von Tabellen zulassen

Das folgende JSON-Richtliniendokument gewährt die Berechtigung zum Erstellen von Tabellen in `myExampleLedger`. Die -`qldb:PartiQLCreateTable`Aktion erfordert Berechtigungen für den Tabellenressourcentyp. Die Tabellen-ID einer neuen Tabelle ist jedoch zum Zeitpunkt der Ausführung einer -`CREATE TABLE`Anweisung nicht bekannt. Daher muss eine Richtlinie, die die `qldb:PartiQLCreateTable` Berechtigung erteilt, einen Platzhalter (*) im Tabellen-ARN verwenden, um die Ressource anzugeben.

Um diese Richtlinie zu verwenden, ersetzen Sie *`us-east-1`*, *`123456789012`* und *`myExampleLedger`* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLCreateTablePermission",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLCreateTable"
            ],
            "Resource": [
```

```
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*"
            ]
        }
    ]
}
```

Erlauben des Zugriffs zum Erstellen von Tabellen basierend auf Anforderungs-Tags

Das folgende JSON-Richtliniendokument verwendet eine Bedingung, die auf dem `aws:RequestTag` Kontextschlüssel basiert, um die Berechtigung zum Erstellen von Tabellen in zu erteilen`myExampleLedger`. Berechtigungen werden nur erteilt, wenn das Anforderungs-Tag den Wert `environment` hat`development`. Das Markieren von Tabellen bei der Erstellung erfordert Zugriff auf die `qldb:TagResource` Aktionen `qldb:PartiQLCreateTable` und . Informationen zum Markieren von Tabellen bei der Erstellung finden Sie unter Markieren von Tabellen.

Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012* und *myExampleLedger* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLCreateTablePermission",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLCreateTable",
                "qldb:TagResource"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*"
            ],
            "Condition": {
                "StringEquals": { "aws:RequestTag/environment": "development" }
            }
        }
    ]
```

```
}
```

Exportieren eines Journals in einen Amazon S3-Bucket

Schritt 1: QLDB-Journalexportberechtigungen

Im folgenden Beispiel erteilen Sie einem Benutzer in Ihren AWS-Konto Berechtigungen, die `qldb:ExportJournalToS3` Aktion für eine QLDB-Ledger-Ressource auszuführen. Sie erteilen auch Berechtigungen zum Ausführen der `iam:PassRole` Aktion für die IAM-Rollenressource, die Sie an den QLDB-Service übergeben möchten. Dies ist für alle Journalexportanforderungen erforderlich.

Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012myExampleLedger*, und *qldb-s3-export* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportPermission",
            "Effect": "Allow",
            "Action": "qldb:ExportJournalToS3",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "IAMPassRolePermission",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/qldb-s3-export",
            "Condition": {
               "StringEquals": {
                  "iam:PassedToService": "qldb.amazonaws.com"
               }
            }
        }
    ]
}
```

Schritt 2: Amazon S3-Bucket-Berechtigungen

Im folgenden Beispiel verwenden Sie eine IAM-Rolle, um QLDB Zugriff zum Schreiben in einen Ihrer Amazon S3-Buckets zu gewähren, *DOC-EXAMPLE-BUCKET*. Dies ist auch für alle QLDB-Journalexporte erforderlich.

Zusätzlich zur Erteilung der `s3:PutObject` Berechtigung gewährt die Richtlinie auch die `s3:PutObjectAcl` Berechtigung zum Festlegen der Zugriffskontrolllisten (ACL)-Berechtigungen für ein Objekt.

Um diese Richtlinie zu verwenden, ersetzen Sie *DOC-EXAMPLE-BUCKET* im Beispiel durch Ihren Amazon S3-Bucket-Namen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

Anschließend fügen Sie diese Berechtigungsrichtlinie an eine IAM-Rolle an, die QLDB annehmen kann, um auf Ihren Amazon S3-Bucket zuzugreifen. Das folgende JSON-Dokument ist ein Beispiel für eine Vertrauensrichtlinie, die es QLDB ermöglicht, die IAM-Rolle nur für jede QLDB-Ressource im Konto zu übernehmen123456789012.

Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1* und *123456789012* im Beispiel durch Ihre eigenen Informationen.

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

```
                    "aws:SourceArn": "arn:aws:qldb:us-east-1:123456789012:*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

Streamen eines Journals zu Kinesis Data Streams

Schritt 1: QLDB-Journal-Stream-Berechtigungen

Im folgenden Beispiel erteilen Sie einem Benutzer in Ihren AWS-Konto Berechtigungen, die `qldb:StreamJournalToKinesis` Aktion für alle QLDB-Stream-Subressourcen in einem Ledger auszuführen. Sie erteilen auch Berechtigungen zum Ausführen der `iam:PassRole` Aktion für die IAM-Rollenressource, die Sie an den QLDB-Service übergeben möchten. Dies ist für alle Journalstream-Anforderungen erforderlich.

Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012*, *myExampleLedger*und *qldb-kinesis-stream* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalStreamPermission",
            "Effect": "Allow",
            "Action": "qldb:StreamJournalToKinesis",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:stream/myExampleLedger/*"
        },
        {
            "Sid": "IAMPassRolePermission",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/qldb-kinesis-stream",
            "Condition": {
               "StringEquals": {
                  "iam:PassedToService": "qldb.amazonaws.com"
               }
            }
```

```
        }
    ]
}
```

Schritt 2: Berechtigungen für Kinesis Data Streams

Im folgenden Beispiel verwenden Sie eine IAM-Rolle, um QLDB Zugriff zum Schreiben von Datensätzen in Ihren Amazon Kinesis Data Stream, , zu gewähren*stream-for-qldb*. Dies ist auch für alle QLDB-Journalstreams erforderlich.

Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012* und *stream-for-qldb* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBStreamKinesisPermissions",
            "Action": [ "kinesis:PutRecord*", "kinesis:DescribeStream",
 "kinesis:ListShards" ],
            "Effect": "Allow",
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-for-qldb"
        }
    ]
}
```

Anschließend fügen Sie diese Berechtigungsrichtlinie einer IAM-Rolle an, die QLDB annehmen kann, um auf Ihren Kinesis-Datenstrom zuzugreifen. Das folgende JSON-Dokument ist ein Beispiel für eine Vertrauensrichtlinie, die es QLDB ermöglicht, eine IAM-Rolle für jeden QLDB-Stream im Konto `myExampleLedger` nur `123456789012` für das Ledger zu übernehmen.

Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012* und *myExampleLedger* im Beispiel durch Ihre eigenen Informationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "qldb.amazonaws.com"
```

```
            },
            "Action": [ "sts:AssumeRole" ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:qldb:us-
east-1:123456789012:stream/myExampleLedger/*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

Aktualisieren von QLDB-Ledgern basierend auf Tags

Sie können Bedingungen in Ihrer identitätsbasierten Richtlinie verwenden, um den Zugriff auf QLDB-Ressourcen basierend auf Tags zu steuern. Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die die Aktualisierung eines Ledgers gestattet. Die Berechtigung wird jedoch nur erteilt, wenn das Ledger-Tag den Wert des Benutzernamens dieses Benutzers `Owner` hat. Diese Richtlinie gewährt auch die Berechtigungen, die für die Ausführung dieser Aktion auf der Konsole erforderlich sind.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListLedgersInConsole",
            "Effect": "Allow",
            "Action": "qldb:ListLedgers",
            "Resource": "*"
        },
        {
            "Sid": "UpdateLedgerIfOwner",
            "Effect": "Allow",
            "Action": "qldb:UpdateLedger",
            "Resource": "arn:aws:qldb:*:*:ledger/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
            }
        }
```

```
    ]
}
```

Sie können diese Richtlinie den -Benutzern in Ihrem Konto zuweisen. Wenn ein Benutzer mit dem Namen `richard-roe` versucht, ein QLDB-Ledger zu aktualisieren, muss das Ledger mit dem Tag `0wner=richard-roe` oder versehen sein`owner=richard-roe`. Andernfalls wird der Zugriff abgelehnt. Der Tag-Schlüssel `0wner` der Bedingung stimmt sowohl mit `0wner` als auch mit `owner` überein, da die Namen von Bedingungsschlüsseln nicht zwischen Groß- und Kleinschreibung unterscheiden. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.

Serviceübergreifende Confused-Deputy-Prävention

Das Confused-Deputy-Problem ist ein Sicherheitsproblem, bei dem eine juristische Stelle, die nicht über die Berechtigung zum Ausführen einer Aktion verfügt, eine privilegiertere juristische Stelle zwingen kann, die Aktion auszuführen. In kann der AWSserviceübergreifende Identitätswechsel zu dem Problem des verwirrten Stellvertreters führen.

Ein dienstübergreifender Identitätswechsel kann auftreten, wenn ein Dienst (der Anruf-Dienst) einen anderen Dienst anruft (den aufgerufenen Dienst). Der Anruf-Dienst kann so manipuliert werden, dass er seine Berechtigungen verwendet, um auf die Ressourcen eines anderen Kunden zu reagieren, auf die er sonst nicht zugreifen dürfte. Um das Problem des verwirrten Stellvertreters zu vermeiden, AWS stellt Tools bereit, mit denen Sie Ihre Daten für alle Services mit Serviceprinzipalen schützen können, die Zugriff auf Ressourcen in Ihrem Konto erhalten haben.

Wir empfehlen die Verwendung der `aws:SourceAccount` globalen Bedingungskontextschlüssel `aws:SourceArn` und in Ressourcenrichtlinien, um die Berechtigungen einzuschränken, die Amazon QLDB einem anderen Service für die Ressource erteilt. Wenn Sie beide globalen Bedingungskontextschlüssel verwenden, müssen der `aws:SourceAccount` Wert und das Konto im `aws:SourceArn` Wert dieselbe Konto-ID verwenden, wenn sie in derselben Richtlinienanweisung verwendet werden.

In der folgenden Tabelle sind mögliche Werte von `aws:SourceArn` für die - `ExportJournalToS3` und `StreamsJournalToKinesis` QLDB-API-Operationen aufgeführt. Diese Operationen sind für dieses Sicherheitsproblem relevant, da sie AWS Security Token Service (AWS STS) aufrufen, um eine von Ihnen angegebene IAM-Rolle anzunehmen.

API-Operation	Aufgerufener Service	aws:SourceArn
`ExportJournalToS3`	AWS STS (AssumeRole)	Ermöglicht QLDB, die Rolle für alle QLDB-Ressourcen im Konto zu übernehmen: `arn:aws:qldb: us-east-1 :123456789012 :*` Derzeit unterstützt QLDB diesen Platzhalter-ARN nur für Journalexporte.

`StreamsJournalToKinesis`	AWS STS (AssumeRole)	Ermöglicht QLDB, die Rolle für einen bestimmten QLDB-Stream zu übernehmen:
		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /IiPT4brpZCqCq3f4MTHbYy`
		Hinweis: Sie können im ARN erst eine Stream-ID angeben, nachdem die Stream-Ressource erstellt wurde. Mit diesem ARN können Sie zulassen, dass die Rolle nur für einen einzelnen QLDB-Stream verwendet wird.
		Ermöglicht QLDB, die Rolle für alle QLDB-Streams eines Ledgers zu übernehmen:
		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /*`
		Ermöglicht QLDB, die Rolle für alle QLDB-Streams im Konto zu übernehmen:
		`arn:aws:qldb: us-east-1 :123456789012 :stream/*`
		Ermöglicht QLDB, die Rolle für alle QLDB-Ressourcen im Konto zu übernehmen:
		`arn:aws:qldb: us-east-1 :123456789012 :*`

Der effektivste Weg, um sich vor dem Confused-Deputy-Problem zu schützen, ist die Verwendung des globalen Bedingungskontext-Schlüssels `aws:SourceArn` mit dem vollständigen ARN der Ressource. Wenn Sie den vollständigen ARN der Ressource nicht kennen oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den `aws:SourceArn` globalen Kontextbedingungsschlüssel mit Platzhalterzeichen (*) für die unbekannten Teile des ARN, z. B. `arn:aws:qldb:us-east-1:`*123456789012*`:*`.

Das folgende Beispiel für eine Vertrauensrichtlinie für eine IAM-Rolle zeigt, wie Sie die `aws:SourceAccount` globalen Bedingungskontextschlüssel `aws:SourceArn` und verwenden können, um das Problem des verwirrten Stellvertreters zu vermeiden. Mit dieser Vertrauensrichtlinie kann QLDB die Rolle für jeden QLDB-Stream im Konto `myExampleLedger` nur `123456789012` für den Ledger übernehmen.

Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012* und *myExampleLedger* im Beispiel durch Ihre eigenen Informationen.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "qldb.amazonaws.com"
    },
    "Action": [ "sts:AssumeRole" ],
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:qldb:us-east-1:123456789012:stream/myExampleLedger/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

AWS Von verwaltete Richtlinien für Amazon QLDB

Eine AWS von verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet wird AWS. Von AWS verwaltete Richtlinien sind so konzipiert, dass sie Berechtigungen für viele häufige Anwendungsfälle bereitstellen, sodass Sie mit der Zuweisung von Berechtigungen für Benutzer, Gruppen und Rollen beginnen können.

Beachten Sie, dass von AWS verwaltete Richtlinien möglicherweise keine Berechtigungen mit den geringsten Berechtigungen für Ihre spezifischen Anwendungsfälle gewähren, da sie für alle - AWS Kunden verfügbar sind. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie kundenverwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in verwalteten AWS Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS von verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert, wirkt sich die Aktualisierung auf alle Prinzipalidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie angefügt ist. aktualisiert am AWS wahrscheinlichsten eine von AWS verwaltete Richtlinie, wenn ein neuer gestartet AWS-Service wird oder neue API-Operationen für vorhandene Services verfügbar werden.

Weitere Informationen finden Sie unter VonAWS verwaltete Richtlinien im IAM-Benutzerhandbuch.

Weitere Informationen zu den QLDB-API-Operationen in diesen AWS verwalteten Richtlinien finden Sie unter Amazon QLDB API-Referenz.

Themen

- AWS Von verwaltete Richtlinie: AmazonQLDBReadOnly
- AWS Von verwaltete Richtlinie: AmazonQLDBFullAccess
- AWS Von verwaltete Richtlinie: AmazonQLDBConsoleFullAccess
- QLDB-Updates für - AWS verwaltete Richtlinien

AWS Von verwaltete Richtlinie: AmazonQLDBReadOnly

Verwenden Sie die AmazonQLDBReadOnly-Richtlinie, um Leseberechtigungen für alle QLDB-Ressourcen zu erteilen. Sie können diese Richtlinie an Ihre IAM-Identitäten anfügen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen für den `qldb` Service.

- Ermöglicht es Prinzipalen, alle QLDB-Ressourcen und ihre Tags zu beschreiben und aufzulisten. Zu diesen Ressourcen gehören Ledger, Amazon S3-Exportaufträge und Streams zu Kinesis Data Streams.
- Ermöglicht es Prinzipalen, einen Block, einen Digest oder eine Revision aus dem Journal in einem beliebigen Ledger abzurufen, um die Daten kryptografisch zu überprüfen.
- Erlaubt es Prinzipalen nicht, PartiQL-Befehle für Tabellen in Ledgern auszuführen.

AWS Von verwaltete Richtlinie: AmazonQLDBFullAccess

Verwenden Sie die AmazonQLDBFullAccess-Richtlinie, um allen QLDB-Ressourcen über die QLDB-API oder die vollständige Administratorberechtigungen zu erteilen AWS CLI. Sie können diese Richtlinie an Ihre IAM-Identitäten anfügen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

- `qldb`
  - Ermöglicht es Prinzipalen, alle QLDB-Ressourcen und ihre Tags zu erstellen, zu beschreiben, aufzulisten und zu verwalten. Zu diesen Ressourcen gehören Ledger, Amazon S3-Exportaufträge und Streams zu Kinesis Data Streams.
  - Ermöglicht es Prinzipalen, alle PartiQL-Befehle für alle Tabellen in einem Ledger mithilfe des QLDB-Treibers oder der QLDB-Shell auszuführen.
  - Ermöglicht es Prinzipalen, einen Block, einen Digest oder eine Revision aus dem Journal in einem beliebigen Ledger abzurufen, um die Daten kryptografisch zu überprüfen.
- `iam` – Ermöglicht es Prinzipalen, jede IAM-Rollenressource in Ihrem Konto an den QLDB-Service zu übergeben. Dies ist für alle Journalexport- und Stream-Anforderungen erforderlich.

AWS Von verwaltete Richtlinie: AmazonQLDBConsoleFullAccess

Verwenden Sie die AmazonQLDBConsoleFullAccess-Richtlinie, um allen QLDB-Ressourcen über die AWS Management Console, die QLDB-API oder die vollständige Administratorberechtigungen zu erteilen AWS CLI. Sie können diese Richtlinie an Ihre IAM-Identitäten anfügen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

- `qldb`
  - Ermöglicht es Prinzipalen, alle QLDB-Ressourcen und ihre Tags zu erstellen, zu beschreiben, aufzulisten und zu verwalten. Zu diesen Ressourcen gehören Ledger, Amazon S3-Exportaufträge und Streams zu Kinesis Data Streams.
  - Ermöglicht es Prinzipalen, alle PartiQL-Befehle für alle Tabellen in einem Ledger mithilfe der QLDB-Konsole, des QLDB-Treibers oder der QLDB-Shell auszuführen.
  - Ermöglicht es Prinzipalen, Beispielanwendungsdaten mithilfe der QLDB-Konsole in jedes Ledger einzufügen.
  - Ermöglicht es Prinzipalen, einen Block, einen Digest oder eine Revision aus dem Journal in einem beliebigen Ledger abzurufen, um die Daten kryptografisch zu überprüfen.
- `dbqms` – Ermöglicht es Prinzipalen, alle Aktionen im Database Query Metadata Service zu verwenden. Dies ist ein interner Service, den die QLDB-Konsole benötigt, um aktuelle und gespeicherte Abfragen für den PartiQL-Abfrage-Editor zu erstellen, zu beschreiben und zu verwalten.
- `kinesis` – Ermöglicht es Prinzipalen, Ressourcen von Amazon Kinesis Data Streams zu beschreiben und aufzulisten. Diese Ressourcen sind die Zielziele, in die QLDB-Stream-Ressourcen Daten schreiben können.
- `iam` – Ermöglicht es Prinzipalen, jede IAM-Rollenressource in Ihrem Konto an den QLDB-Service zu übergeben. Dies ist für alle Journalexport- und Stream-Anforderungen erforderlich.

QLDB-Updates für - AWS verwaltete Richtlinien

Anzeigen von Details zu Aktualisierungen für - AWS verwaltete Richtlinien für QLDB, seit dieser Service mit der Verfolgung dieser Änderungen begonnen hat. Um automatische Warnungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS-Feed auf der Seite QLDB-Versionsverlauf.

Änderung	Beschreibung	Datum
AmazonQLDBFullAccess, AmazonQLDBConsoleFullAccess – Aktualisierung vorhandener Richtlinien	QLDB hat eine neue Berechtigung hinzugefügt, damit Prinzipale Dokumentrevisionen in allen Ledgern	04. November 2022

Änderung	Beschreibung	Datum
	im `STANDARD` Berechtigungsmodus redigieren können.	
AmazonQLDBFullAccess, AmazonQLDBConsoleFullAccess – Aktualisierung vorhandener Richtlinien	QLDB hat neue Berechtigungen hinzugefügt, damit Prinzipale jede IAM-Rollenressource in Ihrem Konto an den QLDB-Service übergeben können. Dies ist für alle Journalexport- und Stream-Anforderungen erforderlich.	2. September 2021
AmazonQLDBReadOnly – Aktualisierung auf eine vorhandene Richtlinie	QLDB hat eine doppelte `qldb:GetBlock` Aktion entfernt, die zuvor zweimal aufgeführt wurde, und das `"Effect"` Feld so neu angeordnet, dass es vor dem `"Action"` Feld erscheint.	1. Juli 2021

Änderung	Beschreibung	Datum
AmazonQLDBFullAccess , AmazonQLDBConsoleFullAccess – Aktualisierung vorhandener Richtlinien	QLDB hat neue Berechtigungen hinzugefügt, damit Prinzipale den Berechtigungsmodus in allen Ledgern aktualisieren und alle PartiQL-Befehle in allen Ledgern im neuen STANDARD Berechtigungsmodus ausführen können. Der STANDARD Berechtigungsmodus unterstützt die Zugriffskontrolle auf Tabellenebene und Granularität für PartiQL-Befehle. Um den neuen Berechtigungsmodus zu erleichtern, führte QLDB eine Reihe von IAM-Aktionen für PartiQL-Befehlstypen und Amazon-Ressourcennamen (ARNs) für QLDB-Tabellenressourcen ein. Diese beiden Richtlinien werden aktualisiert, um die neuen PartiQL-Aktionen aufzunehmen, um vollen Zugriff auf STANDARD Ledger zu gewähren.	27. Mai 2021
QLDB hat mit der Verfolgung von Änderungen begonnen	QLDB hat mit der Verfolgung von Änderungen für seine AWS -verwalteten Richtlinien begonnen.	1. März 2021

Fehlerbehebung für Amazon-QLDB-Identität und -Zugriff

Verwenden Sie die folgenden Informationen, um häufige Probleme zu diagnostizieren und zu beheben, die beim Arbeiten mit QLDB und IAM auftreten können.

Themen

Ich bin nicht autorisiert, eine Aktion in QLDB auszuführen

Wenn Ihnen AWS Management Console mitteilt, dass Sie nicht zur Ausführung einer Aktion autorisiert sind, müssen Sie sich an Ihren Administrator wenden, um Unterstützung zu erhalten. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson`-Benutzer versucht, die Konsole zum Anzeigen von Details zu einer fiktiven *`myExampleLedger`*-Ressource zu verwenden, jedoch nicht über `qldb:`*`DescribeLedger`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 qldb:DescribeLedger on resource: myExampleLedger
```

In diesem Fall bittet Mateo seinen Administrator um die Aktualisierung seiner Richtlinien, um unter Verwendung der Aktion *`myExampleLedger`* auf die Ressource `qldb:`*`DescribeLedger`* zugreifen zu können.

Ich bin nicht autorisiert, iam durchzuführen:PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Ausführen der `iam:PassRole` Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an QLDB übergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine vorhandene Rolle an diesen Service zu übergeben, anstatt eine neue Servicerolle oder serviceverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in QLDB auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Hinweise zur Behebung dieses Fehlers, der für den Journalexport oder die Stream-Operationen spezifisch ist, finden Sie unter Fehlerbehebung für Amazon QLDB.

Ich möchte Personen außerhalb meines AWS-Konto Zugriff auf meine QLDB-Ressourcen gewähren

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Services, die ressourcenbasierte Richtlinien oder Zugriffssteuerungslisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen dazu, ob QLDB diese Funktionen unterstützt, finden Sie unter Funktionsweise von Amazon QLDB mit IAM.
- Informationen zum Gewähren des Zugriffs auf Ihre Ressourcen in Ihrem Besitz finden AWS-Konten Sie unter Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen AWS-Konto , das Sie besitzen im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre -Ressourcen gewähren AWS-Konten, finden Sie unter Gewähren von Zugriff auf im AWS-Konten Besitz von Dritten im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.

- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.

Protokollierung und Überwachung in Amazon QLDB

Überwachung ist wichtig, um die Zuverlässigkeit, Verfügbarkeit und Leistung von Amazon QLDB und Ihrer - AWS Lösungen aufrechtzuerhalten. Sie sollten Überwachungsdaten von allen Teilen Ihrer - AWS Lösung sammeln, damit Sie Ausfälle an mehreren Punkten leichter debuggen können. Bevor Sie jedoch mit der Überwachung von QLDB beginnen, sollten Sie einen Überwachungsplan erstellen, der Antworten auf die folgenden Fragen enthält:

- Was sind Ihre Ziele bei der Überwachung?
- Welche Ressourcen werden überwacht?
- Wie oft werden diese Ressourcen überwacht?
- Welche Überwachungstools werden verwendet?
- Wer soll die Überwachungsaufgaben ausführen?
- Wer soll benachrichtigt werden, wenn Fehler auftreten?

Der nächste Schritt besteht darin, eine Grundlage für die normale QLDB-Leistung in Ihrer Umgebung zu schaffen, indem Sie die Leistung zu verschiedenen Zeiten und unter verschiedenen Lastbedingungen messen. Speichern Sie bei der Überwachung von QLDB historische Überwachungsdaten, damit Sie sie mit aktuellen Leistungsdaten vergleichen, normale Leistungsmuster und Leistungsanomalien identifizieren und Methoden zur Behebung von Problemen entwickeln können.

Zur Festlegung eines Grundwertes sollten Sie mindestens die folgenden Elemente überwachen:

- Lese- und Schreib-E/As und Speicher, damit Sie die Verbrauchsmuster Ihres Ledgers für Fakturierungszwecke nachverfolgen können.
- Befehlslatenz, sodass Sie die Leistung Ihres Ledgers beim Ausführen von Datenoperationen verfolgen können.
- Ausnahmen, damit Sie feststellen können, ob Anfragen zu Fehlern führten.

Themen

Überwachungstools

AWS bietet verschiedene Tools, mit denen Sie Amazon QLDB überwachen können. Sie können einige dieser Tools so konfigurieren, dass diese die Überwachung für Sie übernehmen, während bei anderen Tools ein manuelles Eingreifen nötig ist. Wir empfehlen, dass Sie die Überwachungsaufgaben möglichst automatisieren.

Themen

Automatisierte Überwachungstools

Sie können die folgenden automatisierten Tools zur Überwachung von QLDB verwenden und auftretende Probleme melden:

- Amazon CloudWatch -Alarme – Überwachen Sie eine einzelne Metrik über einen von Ihnen angegebenen Zeitraum und führen Sie eine oder mehrere Aktionen aus, die auf dem Wert der Metrik im Verhältnis zu einem bestimmten Schwellenwert über eine Reihe von Zeiträumen basieren. Die Aktion ist eine Benachrichtigung, die an ein Amazon Simple Notification Service (Amazon SNS)-Thema oder eine Amazon EC2 Auto Scaling-Richtlinie gesendet wird. - CloudWatch Alarme rufen keine Aktionen auf, nur weil sie sich in einem bestimmten Status befinden. Der Status muss geändert und für eine bestimmte Anzahl von Zeiträumen beibehalten worden sein. Weitere Informationen finden Sie unter Überwachung mit Amazon CloudWatch.
- Amazon CloudWatch Logs – Überwachen, Speichern und Zugriff auf Ihre Protokolldateien von AWS CloudTrail oder anderen Quellen. Weitere Informationen finden Sie unter Überwachen von Protokolldateien im Amazon- CloudWatch Benutzerhandbuch.
- Amazon CloudWatch Events – Ordnen Sie Ereignisse zu und leiten Sie sie an eine oder mehrere Zielfunktionen oder Streams weiter, um Änderungen vorzunehmen, Statusinformationen zu erfassen und Korrekturmaßnahmen zu ergreifen. Weitere Informationen finden Sie unter Was ist Amazon CloudWatch Events? im Amazon- CloudWatch Benutzerhandbuch.

- AWS CloudTrail Protokollüberwachung – Teilen Sie Protokolldateien zwischen Konten, überwachen Sie CloudTrail Protokolldateien in Echtzeit, indem Sie sie an - CloudWatch Protokolle senden, schreiben Sie Anwendungen zur Protokollverarbeitung in Java und überprüfen Sie, ob sich Ihre Protokolldateien nach der Bereitstellung durch nicht geändert haben CloudTrail. Weitere Informationen finden Sie unter Arbeiten mit CloudTrail Protokolldateien im AWS CloudTrail - Benutzerhandbuch.

Manuelle Überwachungstools

Ein weiterer wichtiger Bestandteil der Überwachung von QLDB ist die manuelle Überwachung derjenigen Elemente, die die CloudWatch Alarme nicht abdecken. QLDB CloudWatch, Trusted Advisorund andere AWS Management Console Dashboards bieten einen at-a-glance Überblick über den Zustand Ihrer AWS Umgebung. Wir empfehlen Ihnen, auch die Protokolldateien auf Amazon QLDB zu überprüfen.

- Das QLDB-Dashboard zeigt Folgendes:
  - Lese- und Schreib-E/A-Vorgänge
  - Journal und indizierter Speicher
  - Befehlslatenz
  - Ausnahmen
- Auf der - CloudWatch Startseite wird Folgendes angezeigt:
  - Aktuelle Alarme und Status
  - Diagramme mit Alarmen und Ressourcen
  - Servicestatus

  Darüber hinaus können Sie mit Folgendes CloudWatch tun:
  - Erstellen angepasster Dashboards zur Überwachung der gewünschten Services.
  - Aufzeichnen von Metrikdaten, um Probleme zu beheben und Trends zu erkennen
  - Suchen und Durchsuchen all Ihrer AWS Ressourcenmetriken
  - Erstellen und Bearbeiten von Alarmen, um über Probleme benachrichtigt zu werden

Überwachung mit Amazon CloudWatch

Sie können Amazon QLDB mit überwachen CloudWatch, das Rohdaten von Amazon QLDB sammelt und zu lesbaren near-real-time Metriken verarbeitet. Diese Statistiken werden zwei Wochen lang

aufgezeichnet, damit Sie auf Verlaufsinformationen zugreifen können und einen besseren Überblick darüber erhalten, wie Ihre Webanwendung oder der Service ausgeführt werden. Standardmäßig werden QLDB-Metrikdaten CloudWatch automatisch in Abständen von 1 oder 15 Minuten an gesendet. Weitere Informationen finden Sie unter Was sind Amazon CloudWatch, Amazon CloudWatch Events und Amazon CloudWatch Logs? im Amazon- CloudWatch Benutzerhandbuch.

Themen

- Wie verwende ich QLDB-Metriken?
- Amazon-QLDB-Metriken und -Dimensionen
- Erstellen von CloudWatch Alarmen zur Überwachung von Amazon QLDB

Wie verwende ich QLDB-Metriken?

Die von QLDB gemeldeten Metriken liefern Informationen, die Sie auf unterschiedliche Weise analysieren können. In der folgenden Liste finden Sie einige häufige Verwendungszwecke für die Metriken. Es handelt sich dabei um Vorschläge für den Einstieg und nicht um eine umfassende Liste.

- Sie können `JournalStorage` und `IndexedStorage` über einen bestimmten Zeitraum überwachen, um zu verfolgen, wie viel Speicherplatz Ihr Ledger belegt.
- Sie können `ReadIOs` und `WriteIOs` über einem angegebenen Zeitraum überwachen, um nachzuverfolgen, wie viele Anfragen Ihr Ledger verarbeitet.
- Sie können `CommandLatency` überwachen, um die Leistung Ihres Ledgers für Datenoperationen zu verfolgen und die Arten der Befehle zu analysieren, die zur größten Latenz führen.

Amazon-QLDB-Metriken und -Dimensionen

Wenn Sie mit Amazon QLDB interagieren, werden die folgenden Metriken und Dimensionen an gesendet CloudWatch. Speichermetriken werden alle 15 Minuten gemeldet, und alle anderen Metriken werden aggregiert und jede Minute gemeldet. Sie können die folgenden Verfahren verwenden, um die Metriken für QLDB anzuzeigen.

So zeigen Sie Metriken mit der CloudWatch Konsole an

Metriken werden zunächst nach dem Service-Namespace und anschließend nach den verschiedenen Dimensionskombinationen in den einzelnen Namespaces gruppiert.

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.

2. Ändern Sie, falls erforderlich, die Region. Wählen Sie in der Navigationsleiste die Region aus, in der sich Ihre AWS Ressourcen befinden. Weitere Informationen finden Sie unter Regionen und Endpunkte.
3. Wählen Sie im Navigationsbereich Metrics (Metriken) aus.
4. Wählen Sie auf der Registerkarte Alle Metriken die Option QLDB aus.

So zeigen Sie Metriken mit der an AWS CLI

- Geben Sie als Eingabeaufforderung den folgenden Befehl ein.

```
aws cloudwatch list-metrics --namespace "AWS/QLDB"
```

CloudWatch zeigt die folgenden Metriken für QLDB an.

Amazon-QLDB-Dimensionen und -Metriken

Die Metriken und Dimensionen, die Amazon QLDB an Amazon sendet, CloudWatch sind hier aufgeführt.

QLDB-Metriken

Metrik	Beschreibung
`JournalStorage`	Der gesamte Speicherplatz für das Journal des Ledgers, in 15-Minuten-Intervallen gemeldet. Das Journal enthält den vollständigen, unveränderlichen und überprüfbaren Verlauf aller Datenänderungen. Einheiten: `Bytes` Maße: `LedgerName`
`IndexedStorage`	Der gesamte Speicherplatz Tabellen, Indizes und den indizierten Verlauf des Ledgers, in 15-Minuten-Interva llen gemeldet. Der indizierte Speicher besteht aus für Hochleistungs-Abfragen optimierten Ledgerdaten. Einheiten: `Bytes`

Metrik	Beschreibung
	Maße: `LedgerName`
`ReadIOs`	Die Anzahl der Lese-I/O-Anforderungen, die in Intervall en von einer Minute gemeldet werden. Dadurch werden alle Arten von Lesevorgängen erfasst, einschließlich Datentransaktionen, Verifizierungsanforderungen, Journalexporte und Journalstreams. Einheiten: `Count` Maße: `LedgerName`
`WriteIOs`	Die Anzahl der Schreib-I/O-Anforderungen, die in Intervall en von einer Minute gemeldet werden. Einheiten: `Count` Maße: `LedgerName`
`CommandLatency`	Die Zeitdauer für Datenoperationen, gemeldet in 1-Minuten-Intervallen. Einheiten: `Milliseconds` Maße: `CommandType, LedgerName`
`IsImpaired`	Das Flag, das angibt, ob ein Journalstream zu Kinesis Data Streams beeinträchtigt ist, gemeldet in Intervallen von einer Minute. Der Wert „1“ gibt an, dass sich der Stream in beeinträchtigtem Zustand befindet, und „`0`“ gibt an, dass dies nicht so ist. Einheiten: `Boolean` (0 oder 1) Maße: `LedgerName, StreamId`

Metrik	Beschreibung
`OccConflictExceptions`	Die Anzahl der Anfragen an QLDB, die ein generiere n`OccConflictException` . Weitere Informationen zur optimistischen Gleichzeitigkeitskontrolle (OCC) finden Sie unter Amazon QLDB-Nebenläufigkeit von Amazon QLDB-Nebenläufigkeit. Einheiten: `Count`
`Session4xxExceptions`	Die Anzahl der Anfragen an QLDB, die einen HTTP-4xx-Fehler generieren. Einheiten: `Count`
`Session5xxExceptions`	Die Anzahl der Anfragen an QLDB, die einen HTTP-5xx-Fehler generieren. Einheiten: `Count`
`SessionRateExceede dExceptions`	Die Anzahl der Anfragen an QLDB, die einen generiere n`SessionRateExceededException` . Einheiten: `Count`

Dimensionen für QLDB-Metriken

Die Metriken für QLDB werden durch die Werte für das Konto, den Ledger-Namen, die Stream-ID oder den Befehlstyp qualifiziert. Sie können die CloudWatch Konsole verwenden, um QLDB-Daten entlang einer der Dimensionen in der folgenden Tabelle abzurufen.

Dimension	Beschreibung
`LedgerName`	Diese Dimension schränkt die Daten auf einen spezifischen Ledger ein. Dieser Wert kann ein beliebiger Ledger-Name im aktuellen AWS-Region und im aktuellen sein AWS-Konto.
`StreamId`	Diese Dimension schränkt die Daten auf einen spezifischen Journal-Stream ein. Dieser Wert kann eine beliebige Stream-ID

Dimension	Beschreibung
	für einen Ledger im aktuellen AWS-Region und im aktuellen sein AWS-Konto.
`CommandType`	Diese Dimension beschränkt die Daten auf einen der folgenden QLDB-Daten-API-Befehle: • `AbortTransaction` • `CommitTransaction` • `EndSession` • `ExecuteStatement` • `FetchPage` • `StartSession` • `StartTransaction` Informationen dazu, wie QLDB diese Befehle zur Verwaltung von Datenoperationen verwendet, finden Sie unter Sitzungsmanagement mit dem Fahrer.

Erstellen von CloudWatch Alarmen zur Überwachung von Amazon QLDB

Sie können einen Amazon- CloudWatch Alarm erstellen, der eine Amazon Simple Notification Service (Amazon SNS)-Nachricht sendet, wenn sich der Status des Alarms ändert. Ein Alarm überwacht eine Metrik über einen bestimmten, von Ihnen definierten Zeitraum. Der Alarm führt eine oder mehrere Aktionen durch, basierend auf dem Wert der Metrik im Vergleich zu einem bestimmten Schwellenwert in einer Reihe von Zeiträumen. Die Aktion ist eine Benachrichtigung, die an ein Amazon SNS-Thema oder eine Auto Scaling-Richtlinie gesendet wird.

Alarme rufen nur Aktionen für anhaltende Statusänderungen auf. CloudWatch Alarme rufen keine Aktionen auf, nur weil sie sich in einem bestimmten Status befinden. Der Status muss sich geändert haben und für eine festgelegte Anzahl an Zeiträumen aufrechterhalten worden sein.

Weitere Informationen zum Erstellen von CloudWatch Alarmen finden Sie unter Verwenden von Amazon- CloudWatch Alarmen im Amazon- CloudWatch Benutzerhandbuch.

Automatisieren von Amazon QLDB mit - CloudWatch Ereignissen

Mit Amazon CloudWatch Events können Sie Ihr automatisieren AWS-Services und automatisch auf Systemereignisse reagieren, z. B. bei Problemen mit der Anwendungsverfügbarkeit oder Ressourcenänderungen. Ereignisse von AWS-Services werden nahezu in Echtzeit an CloudWatch Ereignisse übermittelt. Sie können einfache Regeln schreiben, um anzugeben, welche Ereignisse für Sie interessant sind und welche automatisierten Aktionen durchgeführt werden sollen, wenn sich für ein Ereignis eine Übereinstimmung mit einer Regel ergibt. Die folgenden Aktionen können beispielsweise automatisch ausgelöst werden:

- Aufrufen einer - AWS Lambda Funktion
- Aufrufen eines Amazon EC2 Run Command
- Weiterleiten des Ereignisses an Amazon Kinesis Data Streams
- Aktivieren eines - AWS Step Functions Zustandsautomaten
- Benachrichtigen eines Amazon SNS-Themas oder einer Amazon SQS-Warteschlange

Amazon QLDB meldet ein Ereignis an CloudWatch Ereignisse, wenn sich der Status einer Ledger-Ressource in Ihren AWS-Konto ändert. Ereignisse werden derzeit nur auf garantierter at-least-once Basis für QLDB-Ledger-Ressourcen ausgegeben.

Im Folgenden finden Sie ein Beispiel für ein Ereignis, das QLDB gemeldet hat, bei dem sich der Status eines Ledgers in geändert hat`DELETING`.

```
{
    "version" : "0",
    "id" : "2f6557eb-e361-54ef-0f9f-99dd9f171c62",
    "detail-type" : "QLDB Ledger State Change",
    "source" : "aws.qldb",
    "account" : "123456789012",
    "time" : "2019-07-24T21:59:17Z",
    "region" : "us-east-1",
    "resources" : ["arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger"],
    "detail" : {
        "ledgerName" : "exampleLedger",
        "state" : "DELETING"
    }
}
```

Einige Beispiele für die Verwendung von CloudWatch Ereignissen mit QLDB können unter anderem die folgenden sein:

- Aktivieren einer Lambda-Funktion, wenn ein neuer Ledger zunächst im -`CREATING`Zustand erstellt wird und schließlich wird`ACTIVE`.
- Benachrichtigen eines Amazon SNS-Themas, wenn sich der Status Ihres Ledgers in `DELETING` und dann in ändert`DELETED`.

Weitere Informationen finden Sie im Amazon CloudWatch Events-Benutzerhandbuch.

Protokollieren von Amazon-QLDB-API-Aufrufen mit AWS CloudTrail

Amazon QLDB ist in integriert, einem Service AWS CloudTrail, der die Aktionen eines Benutzers, einer Rolle oder eines AWS-Service in QLDB aufzeichnet. CloudTrail erfasst alle API-Aufrufe für die Ressourcenverwaltung für QLDB als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der QLDB-Konsole und Codeaufrufe der QLDB-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon Simple Storage Service (Amazon S3)-Bucket aktivieren, einschließlich Ereignissen für QLDB. Auch wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse in der CloudTrail -Konsole in Event history (Ereignisverlauf) anzeigen. Anhand der von CloudTrailgesammelten Informationen können Sie die an QLDB gestellte Anfrage, die IP-Adresse, von der die Anfrage gestellt wurde, den Initiator der Anfrage, den Zeitpunkt der Anfrage und zusätzliche Details bestimmen.

Weitere Informationen zu CloudTrail, einschließlich Konfiguration und Aktivierung, finden Sie im AWS CloudTrail -Benutzerhandbuch.

QLDB-Informationen in CloudTrail

CloudTrail wird beim Erstellen des Kontos AWS-Konto auf Ihrem aktiviert. Wenn die unterstützte Aktivität in QLDB auftritt, wird diese Aktivität in einem CloudTrail Ereignis zusammen mit anderen AWS-Service Ereignissen im Ereignisverlauf aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem anzeigen, suchen und herunterladen AWS-Konto. Weitere Informationen finden Sie unter Anzeigen von Ereignissen mit dem CloudTrail Ereignisverlauf .

Erstellen Sie für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS-Konto, einschließlich Ereignissen für QLDB, einen Trail. Ein Trail ermöglicht CloudTrail die Bereitstellung von Protokolldateien an einen Amazon S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen in der - AWS Partition und stellt die Protokolldateien in dem von Ihnen angegebenen Amazon S3-Bucket

bereit. Darüber hinaus können Sie andere konfigurieren, AWS-Services um die in den CloudTrail Protokollen erfassten Ereignisdaten weiter zu analysieren und entsprechend zu agieren.

Weitere Informationen finden Sie in folgenden Themen im AWS CloudTrail -Benutzerhandbuch:

- Übersicht zum Erstellen eines Trails
- CloudTrail Von unterstützte Services und Integrationen
- Konfigurieren von Amazon SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen
- Empfangen von CloudTrail Protokolldateien von mehreren Konten

Alle API-Aktionen für QLDB-Ressourcenverwaltung und nicht-transaktionale Daten werden von protokolliert CloudTrail und sind in der Amazon-QLDB-API-Referenz dokumentiert. Zum Beispiel werden durch Aufrufe der `CreateLedger`-, `DescribeLedger`- und `DeleteLedger`-Aktionen Einträge in den CloudTrail -Protokolldateien generiert.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder -Benutzeranmeldeinformationen ausgeführt wurde.
- Ob die Anfrage mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen föderierten Benutzer ausgeführt wurde
- Ob die Anforderung von einem anderen gestellt wurde AWS-Service

Weitere Informationen finden Sie unter CloudTrail -Element userIdentity.

Grundlegendes zu QLDB-Protokolldateieinträgen

Ein Trail ist eine Konfiguration, die die Bereitstellung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, das Datum und die Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordnetes Stacktrace der öffentlichen API-Aufrufe und erscheinen daher nicht in einer bestimmten Reihenfolge.

Das folgende Beispiel zeigt einen - CloudTrail Protokolleintrag, der diese Aktionen demonstriert:

- `CreateLedger`

- DescribeLedger
- ListTagsForResource
- TagResource
- UntagResource
- ListLedgers
- GetDigest
- GetBlock
- GetRevision
- ExportJournalToS3
- DescribeJournalS3Export
- ListJournalS3ExportsForLedger
- ListJournalS3Exports
- DeleteLedger


```
{
  "endTime": 1561497717208,
  "startTime": 1561497687254,
  "calls": [
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:27Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "CreateLedger",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "Name": "CloudtrailTest",
          "PermissionsMode": "ALLOW_ALL"
        },
        "responseElements": {
          "CreationDateTime": 1.561497687403E9,
          "Arn": "arn:aws:qldb:us-east-2:123456789012:ledger/CloudtrailTest",
          "State": "CREATING",
          "Name": "CloudtrailTest"
```

```
        },
        "requestID": "3135aec7-978f-11e9-b313-1dd92a14919e",
        "eventID": "bf703ff9-676f-41dd-be6f-5f666c9f7852",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:27Z\",\"eventSource
\":\"qldb.amazonaws.com\",\"eventName\":\"CreateLedger\",\"awsRegion\":\"us-
east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-
sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08
 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":
{\"Name\":\"CloudtrailTest\",\"PermissionsMode\":\"ALLOW_ALL\"},\"responseElements
\":{\"CreationDateTime\":1.561497687403E9,\"Arn\":\"arn:aws:qldb:us-
east-2:123456789012:ledger/CloudtrailTest\",\"State\":\"CREATING\",\"Name\":
\"CloudtrailTest\"},\"requestID\":\"3135aec7-978f-11e9-b313-1dd92a14919e\",\"eventID\":
\"bf703ff9-676f-41dd-be6f-5f666c9f7852\",\"readOnly\":false,\"eventType\":\"AwsApiCall
\",\"recipientAccountId\":\"123456789012\"}",
      "name": "CreateLedger",
      "request": [
        "com.amazonaws.services.qldb.model.CreateLedgerRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
          "tags": null,
          "permissionsMode": "ALLOW_ALL"
        }
      ],
      "requestId": "3135aec7-978f-11e9-b313-1dd92a14919e"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:43Z",
```

```
        "eventSource": "qldb.amazonaws.com",
        "eventName": "DescribeLedger",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "name": "CloudtrailTest"
        },
        "responseElements": null,
        "requestID": "3af51ba0-978f-11e9-8ae6-837dd17a19f8",
        "eventID": "be128e61-3e38-4503-83de-49fdc7fc0afb",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity
\":{\"type\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-
user\",\"arn\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",
\"accountId\":\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",
\"sessionContext\":{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate
\":\"2019-06-25T21:21:25Z\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":
\"AKIAIOSFODNN7EXAMPLE\",\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId
\":\"123456789012\",\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:43Z
\",\"eventSource\":\"qldb.amazonaws.com\",\"eventName\":\"DescribeLedger\",
\"awsRegion\":\"us-east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":
\"aws-internal/3 aws-sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-
Bit_Server_VM/25.202-b08 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation
\",\"requestParameters\":{\"name\":\"CloudtrailTest\"},\"responseElements
\":null,\"requestID\":\"3af51ba0-978f-11e9-8ae6-837dd17a19f8\",\"eventID\":
\"be128e61-3e38-4503-83de-49fdc7fc0afb\",\"readOnly\":true,\"eventType\":\"AwsApiCall
\",\"recipientAccountId\":\"123456789012\"}",
      "name": "DescribeLedger",
      "request": [
        "com.amazonaws.services.qldb.model.DescribeLedgerRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest"
        }
      ],
      "requestId": "3af51ba0-978f-11e9-8ae6-837dd17a19f8"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
```

```
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:44Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "TagResource",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "resourceArn": "arn%3Aaws%3Aqldb%3Aus-east-2%3A123456789012%3Aledger
%2FCloudtrailTest",
          "Tags": {
            "TagKey": "TagValue"
          }
        },
        "responseElements": null,
        "requestID": "3b1d6371-978f-11e9-916c-b7d64ec76521",
        "eventID": "6101c94a-7683-4431-812b-9a91afb8c849",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:44Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"TagResource\",\"awsRegion\":\"us-east-2\",
\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575
 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202
 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"resourceArn\":\"arn
%3Aaws%3Aqldb%3Aus-east-2%3A123456789012%3Aledger%2FCloudtrailTest\",\"Tags\":{\"TagKey
\":\"TagValue\"}},\"responseElements\":null,\"requestID\":\"3b1d6371-978f-11e9-916c-
b7d64ec76521\",\"eventID\":\"6101c94a-7683-4431-812b-9a91afb8c849\",\"readOnly\":false,
\"eventType\":\"AwsApiCall\",\"recipientAccountId\":\"123456789012\"}",
      "name": "TagResource",
      "request": [
        "com.amazonaws.services.qldb.model.TagResourceRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "resourceArn": "arn:aws:qldb:us-east-2:123456789012:ledger/CloudtrailTest",
```

```
          "tags": {
            "TagKey": "TagValue"
          }
        }
      ],
      "requestId": "3b1d6371-978f-11e9-916c-b7d64ec76521"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:44Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "ListTagsForResource",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "resourceArn": "arn%3Aaws%3Aqldb%3Aus-east-2%3A123456789012%3Aledger
%2FCloudtrailTest"
        },
        "responseElements": null,
        "requestID": "3b56c321-978f-11e9-8527-2517d5bfa8fd",
        "eventID": "375e57d7-cf94-495a-9a48-ac2192181c02",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity
\":{\"type\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-
user\",\"arn\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",
\"accountId\":\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",
\"sessionContext\":{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate
\":\"2019-06-25T21:21:25Z\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId
\":\"AKIAIOSFODNN7EXAMPLE\",\"arn\":\"arn:aws:iam::123456789012:role/Admin
\",\"accountId\":\"123456789012\",\"userName\":\"Admin\"}}},\"eventTime\":
\"2019-06-25T21:21:44Z\",\"eventSource\":\"qldb.amazonaws.com\",\"eventName
\":\"ListTagsForResource\",\"awsRegion\":\"us-east-2\",\"sourceIPAddress\":
\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575 Mac_OS_X/10.13.6
 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202 kotlin/1.3.21 vendor/
Oracle_Corporation\",\"requestParameters\":{\"resourceArn\":\"arn%3Aaws%3Aqldb
%3Aus-east-2%3A123456789012%3Aledger%2FCloudtrailTest\"},\"responseElements\":null,
\"requestID\":\"3b56c321-978f-11e9-8527-2517d5bfa8fd\",\"eventID\":\"375e57d7-
```

```
cf94-495a-9a48-ac2192181c02\",\"readOnly\":true,\"eventType\":\"AwsApiCall\",
\"recipientAccountId\":\"123456789012\"}",
      "name": "ListTagsForResource",
      "request": [
        "com.amazonaws.services.qldb.model.ListTagsForResourceRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "resourceArn": "arn:aws:qldb:us-east-2:123456789012:ledger/CloudtrailTest"
        }
      ],
      "requestId": "3b56c321-978f-11e9-8527-2517d5bfa8fd"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:44Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "UntagResource",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "tagKeys": "TagKey",
          "resourceArn": "arn%3Aaws%3Aqldb%3Aus-east-2%3A123456789012%3Aledger
%2FCloudtrailTest"
        },
        "responseElements": null,
        "requestID": "3b87e59b-978f-11e9-8b9a-bb6dc3a800a9",
        "eventID": "bcdcdca3-699f-4363-b092-88242780406f",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:44Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"UntagResource\",\"awsRegion\":\"us-east-2\",
```

\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575
 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202
 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"tagKeys\":
\"TagKey\",\"resourceArn\":\"arn%3Aaws%3Aqldb%3Aus-east-2%3A123456789012%3Aledger
%2FCloudtrailTest\"},\"responseElements\":null,\"requestID\":\"3b87e59b-978f-11e9-8b9a-
bb6dc3a800a9\",\"eventID\":\"bcdcdca3-699f-4363-b092-88242780406f\",\"readOnly\":false,
\"eventType\":\"AwsApiCall\",\"recipientAccountId\":\"123456789012\"}",
      "name": "UntagResource",
      "request": [
        "com.amazonaws.services.qldb.model.UntagResourceRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "resourceArn": "arn:aws:qldb:us-east-2:123456789012:ledger/CloudtrailTest",
          "tagKeys": [
            "TagKey"
          ]
        }
      ],
      "requestId": "3b87e59b-978f-11e9-8b9a-bb6dc3a800a9"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:44Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "ListLedgers",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": null,
        "responseElements": null,
        "requestID": "3bafb877-978f-11e9-a6de-dbe6464b9dec",
        "eventID": "6ebe7d49-af59-4f29-aaa2-beffe536e20c",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z

\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:44Z\",\"eventSource
\":\"qldb.amazonaws.com\",\"eventName\":\"ListLedgers\",\"awsRegion\":\"us-
east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-
sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08
 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":null,
\"responseElements\":null,\"requestID\":\"3bafb877-978f-11e9-a6de-dbe6464b9dec\",
\"eventID\":\"6ebe7d49-af59-4f29-aaa2-beffe536e20c\",\"readOnly\":true,\"eventType\":
\"AwsApiCall\",\"recipientAccountId\":\"123456789012\"}",

```
      "name": "ListLedgers",
      "request": [
        "com.amazonaws.services.qldb.model.ListLedgersRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "maxResults": null,
          "nextToken": null
        }
      ],
      "requestId": "3bafb877-978f-11e9-a6de-dbe6464b9dec"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:49Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "GetDigest",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "name": "CloudtrailTest"
        },
        "responseElements": null,
        "requestID": "3cddd8a1-978f-11e9-a6de-dbe6464b9dec",
        "eventID": "a5cb60db-e6c5-4f5e-a5fc-0712249622b3",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
```

\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:49Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"GetDigest\",\"awsRegion\":\"us-east-2\",
\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575
 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202
 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"name\":
\"CloudtrailTest\"},\"responseElements\":null,\"requestID\":\"3cddd8a1-978f-11e9-a6de-
dbe6464b9dec\",\"eventID\":\"a5cb60db-e6c5-4f5e-a5fc-0712249622b3\",\"readOnly\":true,
\"eventType\":\"AwsApiCall\",\"recipientAccountId\":\"123456789012\"}",
      "name": "GetDigest",
      "request": [
        "com.amazonaws.services.qldb.model.GetDigestRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
          "digestTipAddress": null
        }
      ],
      "requestId": "3cddd8a1-978f-11e9-a6de-dbe6464b9dec"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:50Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "GetBlock",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "BlockAddress": {
            "IonText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:0}"
          },
          "name": "CloudtrailTest",
          "DigestTipAddress": {
            "IonText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:0}"
          }
        },

```
        "responseElements": null,
        "requestID": "3eaea09f-978f-11e9-bdc2-c1e55368155e",
        "eventID": "1f7da83f-d829-4e35-953d-30b925ceee66",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:50Z\",\"eventSource
\":\"qldb.amazonaws.com\",\"eventName\":\"GetBlock\",\"awsRegion\":\"us-east-2\",
\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575
 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202
 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"BlockAddress
\":{\"IonText\":\"{strandId:\\\"2P2nsG3K2RwHQccUbnAMAj\\\",sequenceNo:0}\"},
\"name\":\"CloudtrailTest\",\"DigestTipAddress\":{\"IonText\":\"{strandId:
\\\"2P2nsG3K2RwHQccUbnAMAj\\\",sequenceNo:0}\"}},\"responseElements\":null,
\"requestID\":\"3eaea09f-978f-11e9-bdc2-c1e55368155e\",\"eventID\":\"1f7da83f-
d829-4e35-953d-30b925ceee66\",\"readOnly\":true,\"eventType\":\"AwsApiCall\",
\"recipientAccountId\":\"123456789012\"}",
      "name": "GetBlock",
      "request": [
        "com.amazonaws.services.qldb.model.GetBlockRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
          "blockAddress": {
            "ionText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:0}"
          },
          "digestTipAddress": {
            "ionText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:0}"
          }
        }
      ],
      "requestId": "3eaea09f-978f-11e9-bdc2-c1e55368155e"
    },
    {
      "cloudtrailEvent": {
```

```
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:55Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "GetRevision",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "BlockAddress": {
            "IonText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:1}"
          },
          "name": "CloudtrailTest",
          "DocumentId": "8UyXvDw6ApoFfVOA2HPfUE",
          "DigestTipAddress": {
            "IonText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:1}"
          }
        },
        "responseElements": null,
        "requestID": "41e19139-978f-11e9-aaed-dfe1dafe37ab",
        "eventID": "43bf2661-5046-41ec-a1d3-87706954aa10",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:55Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"GetRevision\",\"awsRegion\":\"us-east-2\",
\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575
 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202
 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"BlockAddress
\":{\"IonText\":\"{strandId:\\\"2P2nsG3K2RwHQccUbnAMAj\\\",sequenceNo:1}\"},\"name
\":\"CloudtrailTest\",\"DocumentId\":\"8UyXvDw6ApoFfVOA2HPfUE\",\"DigestTipAddress
\":{\"IonText\":\"{strandId:\\\"2P2nsG3K2RwHQccUbnAMAj\\\",sequenceNo:1}\"}},
\"responseElements\":null,\"requestID\":\"41e19139-978f-11e9-aaed-dfe1dafe37ab\",
\"eventID\":\"43bf2661-5046-41ec-a1d3-87706954aa10\",\"readOnly\":true,\"eventType\":
\"AwsApiCall\",\"recipientAccountId\":\"123456789012\"}",
      "name": "GetRevision",
```

```
      "request": [
        "com.amazonaws.services.qldb.model.GetRevisionRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
          "blockAddress": {
            "ionText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:1}"
          },
          "documentId": "8UyXvDw6ApoFfVOA2HPfUE",
          "digestTipAddress": {
            "ionText": "{strandId:\"2P2nsG3K2RwHQccUbnAMAj\",sequenceNo:1}"
          }
        }
      ],
      "requestId": "41e19139-978f-11e9-aaed-dfe1dafe37ab"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:56Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "ExportJournalToS3",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "InclusiveStartTime": 1.561497687254E9,
          "name": "CloudtrailTest",
          "S3ExportConfiguration": {
            "Bucket": "cloudtrailtests-123456789012-us-east-2",
            "Prefix": "CloudtrailTestsJournalExport",
            "EncryptionConfiguration": {
              "ObjectEncryptionType": "SSE_S3"
            }
          },
          "ExclusiveEndTime": 1.561497715795E9
        },
        "responseElements": {
          "ExportId": "BabQhsmJRYDCGMnA2xYBDG"
        },
        "requestID": "423815f8-978f-11e9-afcf-55f7d0f3583d",
        "eventID": "1b5abdc4-52fa-435f-857e-8995ef7a19b7",
```

```
        "readOnly": false,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:56Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"ExportJournalToS3\",\"awsRegion\":\"us-east-2\",
\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-sdk-java/1.11.575
 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08 java/1.8.0_202
 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"InclusiveStartTime
\":1.561497687254E9,\"name\":\"CloudtrailTest\",\"S3ExportConfiguration\":{\"Bucket\":
\"cloudtrailtests-123456789012-us-east-2\",\"Prefix\":\"CloudtrailTestsJournalExport\",
\"EncryptionConfiguration\":{\"ObjectEncryptionType\":\"SSE_S3\"}},\"ExclusiveEndTime
\":1.561497715795E9},\"responseElements\":{\"ExportId\":\"BabQhsmJRYDCGMnA2xYBDG
\"},\"requestID\":\"423815f8-978f-11e9-afcf-55f7d0f3583d\",\"eventID\":
\"1b5abdc4-52fa-435f-857e-8995ef7a19b7\",\"readOnly\":false,\"eventType\":\"AwsApiCall
\",\"recipientAccountId\":\"123456789012\"}",
      "name": "ExportJournalToS3",
      "request": [
        "com.amazonaws.services.qldb.model.ExportJournalToS3Request",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
          "inclusiveStartTime": 1561497687254,
          "exclusiveEndTime": 1561497715795,
          "s3ExportConfiguration": {
            "bucket": "cloudtrailtests-123456789012-us-east-2",
            "prefix": "CloudtrailTestsJournalExport",
            "encryptionConfiguration": {
              "objectEncryptionType": "SSE_S3",
              "kmsKeyArn": null
            }
          }
        }
      ],
      "requestId": "423815f8-978f-11e9-afcf-55f7d0f3583d"
    },
```

```
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:56Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "DescribeJournalS3Export",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "name": "CloudtrailTest",
          "exportId": "BabQhsmJRYDCGMnA2xYBDG"
        },
        "responseElements": null,
        "requestID": "427ebbbc-978f-11e9-8888-e9894c9c4bb9",
        "eventID": "ca8ffc88-16ff-45f5-9042-d94fadb389c3",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:56Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"DescribeJournalS3Export\",\"awsRegion\":
\"us-east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3
 aws-sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08
 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":{\"name
\":\"CloudtrailTest\",\"exportId\":\"BabQhsmJRYDCGMnA2xYBDG\"},\"responseElements
\":null,\"requestID\":\"427ebbbc-978f-11e9-8888-e9894c9c4bb9\",\"eventID\":
\"ca8ffc88-16ff-45f5-9042-d94fadb389c3\",\"readOnly\":true,\"eventType\":\"AwsApiCall
\",\"recipientAccountId\":\"123456789012\"}",
      "name": "DescribeJournalS3Export",
      "request": [
        "com.amazonaws.services.qldb.model.DescribeJournalS3ExportRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
```

```
          "exportId": "BabQhsmJRYDCGMnA2xYBDG"
        }
      ],
      "requestId": "427ebbbc-978f-11e9-8888-e9894c9c4bb9"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:56Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "ListJournalS3ExportsForLedger",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "name": "CloudtrailTest"
        },
        "responseElements": null,
        "requestID": "429ca40c-978f-11e9-8c4b-d13a8018a286",
        "eventID": "34f0e76b-58a5-45be-881c-786d22e34e96",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:56Z\",\"eventSource
\":\"qldb.amazonaws.com\",\"eventName\":\"ListJournalS3ExportsForLedger\",
\"awsRegion\":\"us-east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":
\"aws-internal/3 aws-sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-
Bit_Server_VM/25.202-b08 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation
\",\"requestParameters\":{\"name\":\"CloudtrailTest\"},\"responseElements
\":null,\"requestID\":\"429ca40c-978f-11e9-8c4b-d13a8018a286\",\"eventID\":
\"34f0e76b-58a5-45be-881c-786d22e34e96\",\"readOnly\":true,\"eventType\":\"AwsApiCall
\",\"recipientAccountId\":\"123456789012\"}",
      "name": "ListJournalS3ExportsForLedger",
      "request": [
        "com.amazonaws.services.qldb.model.ListJournalS3ExportsForLedgerRequest",
```

```
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest",
          "maxResults": null,
          "nextToken": null
        }
      ],
      "requestId": "429ca40c-978f-11e9-8c4b-d13a8018a286"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:56Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "ListJournalS3Exports",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": null,
        "responseElements": null,
        "requestID": "42cc1814-978f-11e9-befb-f5dbaa142118",
        "eventID": "4c24d7d6-810c-4cf4-884e-00482278b6ce",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:56Z\",\"eventSource\":
\"qldb.amazonaws.com\",\"eventName\":\"ListJournalS3Exports\",\"awsRegion\":\"us-
east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-
sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08
 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters\":null,
\"responseElements\":null,\"requestID\":\"42cc1814-978f-11e9-befb-f5dbaa142118\",
\"eventID\":\"4c24d7d6-810c-4cf4-884e-00482278b6ce\",\"readOnly\":true,\"eventType\":
\"AwsApiCall\",\"recipientAccountId\":\"123456789012\"}",
      "name": "ListJournalS3Exports",
```

```
      "request": [
        "com.amazonaws.services.qldb.model.ListJournalS3ExportsRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "maxResults": null,
          "nextToken": null
        }
      ],
      "requestId": "42cc1814-978f-11e9-befb-f5dbaa142118"
    },
    {
      "cloudtrailEvent": {
        "userIdentity": {
          "arn": "arn:aws:sts::123456789012:assumed-role/Admin/test-user"
        },
        "eventTime": "2019-06-25T21:21:57Z",
        "eventSource": "qldb.amazonaws.com",
        "eventName": "DeleteLedger",
        "awsRegion": "us-east-2",
        "errorCode": null,
        "requestParameters": {
          "name": "CloudtrailTest"
        },
        "responseElements": null,
        "requestID": "42f439b9-978f-11e9-8b2c-69ef598d66e9",
        "eventID": "429f5163-cba5-4d86-bd7e-f606e057c6cf",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
      },
      "rawCloudtrailEvent": "{\"eventVersion\":\"1.05\",\"userIdentity\":{\"type
\":\"AssumedRole\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE:test-user\",\"arn
\":\"arn:aws:sts::123456789012:assumed-role/Admin/test-user\",\"accountId\":
\"123456789012\",\"accessKeyId\":\"AKIAI44QH8DHBEXAMPLE\",\"sessionContext\":
{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":\"2019-06-25T21:21:25Z
\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AKIAIOSFODNN7EXAMPLE\",
\"arn\":\"arn:aws:iam::123456789012:role/Admin\",\"accountId\":\"123456789012\",
\"userName\":\"Admin\"}}},\"eventTime\":\"2019-06-25T21:21:57Z\",\"eventSource
\":\"qldb.amazonaws.com\",\"eventName\":\"DeleteLedger\",\"awsRegion\":\"us-
east-2\",\"sourceIPAddress\":\"192.0.2.01\",\"userAgent\":\"aws-internal/3 aws-
sdk-java/1.11.575 Mac_OS_X/10.13.6 Java_HotSpot(TM)_64-Bit_Server_VM/25.202-b08
 java/1.8.0_202 kotlin/1.3.21 vendor/Oracle_Corporation\",\"requestParameters
\":{\"name\":\"CloudtrailTest\"},\"responseElements\":null,\"requestID\":
```

```
\"42f439b9-978f-11e9-8b2c-69ef598d66e9\",\"eventID\":\"429f5163-cba5-4d86-bd7e-
f606e057c6cf\",\"readOnly\":false,\"eventType\":\"AwsApiCall\",\"recipientAccountId\":
\"123456789012\"}",
      "name": "DeleteLedger",
      "request": [
        "com.amazonaws.services.qldb.model.DeleteLedgerRequest",
        {
          "customRequestHeaders": null,
          "customQueryParameters": null,
          "name": "CloudtrailTest"
        }
      ],
      "requestId": "42f439b9-978f-11e9-8b2c-69ef598d66e9"
    }
  ]
}
```


Compliance-Validierung für Amazon QLDB

Externe Prüfer bewerten im Rahmen verschiedener - AWS Compliance-Programme die Sicherheit und Compliance von Amazon QLDB, einschließlich, aber nicht beschränkt auf Folgendes:

- System and Organization Controls (SOC)
- Payment Card Industry (PCI)
- International Organization for Standardization (ISO)
- Sicherheitsmanagement und Bewertungsprogramm für Informationssysteme (ISMAP)
- Health Insurance Portability and Accountability Act (HIPAA)

> **Note**
>
> Dies ist keine umfassende Liste von Amazon-QLDB-Zertifizierungen.

Informationen darüber, ob ein in den Geltungsbereich bestimmter Compliance-Programme AWS-Service fällt, finden Sie AWS-Services unter im Geltungsbereich nach Compliance-Programm und wählen Sie das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme

Sie können Auditberichte von Drittanbietern mit herunterladen AWS Artifact. Weitere Informationen finden Sie unter Herunterladen von Berichten unter AWS Artifact .

Ihre Compliance-Verantwortung bei der Verwendung von AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Unterstützung der Compliance bereit:

- Schnellstartanleitungen für Sicherheit und Compliance – In diesen Bereitstellungsleitfäden werden Überlegungen zur Architektur erörtert und Schritte für die Bereitstellung von Basisumgebungen in bereitgestellt AWS , die sich auf Sicherheit und Compliance konzentrieren.
- Architekturerstellung für HIPAA-Sicherheit und -Compliance in Amazon Web Services – In diesem Whitepaper wird beschrieben, wie Unternehmen mithilfe AWS von HIPAA-berechtigte Anwendungen erstellen können.

  > **Note**
  >
  > Nicht alle AWS-Services sind HIPAA-berechtigt. Weitere Informationen finden Sie in der Referenz für HIPAA-berechtigte Services.

- AWS Compliance-Ressourcen – Diese Sammlung von Arbeitsmappen und Leitfäden könnte für Ihre Branche und Ihren Standort gelten.
- AWS Kunden-Compliance-Leitfäden – Verstehen Sie das Modell der geteilten Verantwortung anhand der Berücksichtigung der Compliance. Die Leitfäden fassen die bewährten Methoden zur Sicherung zusammen AWS-Services und ordnen die Leitlinien den Sicherheitskontrollen in mehreren Frameworks zu (einschließlich National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Officer (PCI) und International Organization for Standardization (ISO)).
- Bewertung von Ressourcen mit Regeln im -AWS Config Entwicklerhandbuch – Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub – Dies AWS-Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus innerhalb von AWS. Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Eine Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerungsreferenz.

- AWS Audit Manager – Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um den Umgang mit Risiken und die Einhaltung von Branchenstandards zu vereinfachen.

Ausfallsicherheit in Amazon QLDB

Die AWS globale -Infrastruktur ist um AWS-Regionen und Availability Zones herum aufgebaut. AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die mit einem Netzwerk mit niedriger Latenz, hohem Durchsatz und hoher Redundanz verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Availability Zones ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS-Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Speicherbeständigkeit

QLDB-Journalspeicherfunktionen synchrone Replikation in mehrere Availability Zones bei Transaktions-Commits. Dadurch wird sichergestellt, dass selbst bei einem vollständigen Availability Zone-Ausfall des Journalspeichers die Datenintegrität oder die Fähigkeit zur Aufrechterhaltung eines aktiven Diensts nicht beeinträchtigt ist. Darüber hinaus bietet das QLDB-Journal asynchrone Archive für fehlertoleranten Speicher. Dieses Feature unterstützt die Notfallwiederherstellung im höchst unwahrscheinlichen Fall eines gleichzeitigen Speicherausfalls für mehrere Availability Zones.

QLDB-indizierter Speicher wird durch Replikation in mehrere Availability Zones unterstützt. Dadurch wird sichergestellt, dass selbst bei einem vollständigen Availability Zone-Ausfall des indizierten Speichers die Datenintegrität oder die Fähigkeit zur Aufrechterhaltung eines aktiven Diensts nicht beeinträchtigt ist.

Datenbeständigkeitsfunktionen

Zusätzlich zur AWS globalen -Infrastruktur stellt QLDB die folgenden Funktionen bereit, um Ihren Anforderungen an Ausfallsicherheit und Datensicherung gerecht zu werden.

QLDB-Service-Features

On-Demand-Journal-Export

QLDB bietet eine On-Demand-Journalexportfunktion. Greifen Sie auf den Inhalt Ihres Journals zu, indem Sie Journalblöcke aus Ihrem Ledger in einen Amazon S3-Bucket exportieren. Sie können diese Daten für verschiedene Zwecke wie Datenaufbewahrung, Analyse und Prüfung verwenden. Weitere Informationen finden Sie unter Exportieren von Journaldaten aus Amazon QLDB.

Backup und Wiederherstellung

Die automatische Wiederherstellung für Exporte wird derzeit nicht unterstützt. Exporte stellen eine grundlegende Möglichkeit dar, um eine zusätzliche Datenredundanz mit einer von Ihnen festgelegten Frequenz herzustellen. Ein Wiederherstellungsszenario ist jedoch anwendungsabhängig und die exportierten Datensätze werden in der Regel in einen neuen Ledger geschrieben, wobei die gleiche oder eine ähnliche Erfassungsmethode verwendet wird.

QLDB bietet derzeit keine dedizierte Backup- und zugehörige Wiederherstellungsfunktion.

Journal-Streams

QLDB bietet auch eine kontinuierliche Journalstream-Funktion. Sie können QLDB-Journalstreams in die Amazon Kinesis Streaming-Plattform integrieren, um Journaldaten in Echtzeit zu verarbeiten. Weitere Informationen finden Sie unter Streaming von Journaldaten aus Amazon QLDB.

QLDB-Design-Feature

QLDB ist so konzipiert, dass es widerstandsfähig gegenüber logischer Beschädigung ist. Das QLDB-Journal ist unveränderlich und stellt sicher, dass alle festgeschriebenen Transaktionen im Journal gespeichert werden. Darüber hinaus wird jede übergebene Dokumentänderung aufgezeichnet, da dies die point-in-time Sichtbarkeit aller unbeabsichtigten Änderungen an Ledger-Daten ermöglicht.

QLDB bietet derzeit kein automatisches Wiederherstellungsfeature für logische Beschädigungsszenarien.

Infrastruktursicherheit in Amazon QLDB

Als verwalteter Service ist Amazon QLDB durch die AWS globalen Verfahren zur Gewährleistung der Netzwerksicherheit von geschützt, die im Whitepaper Amazon Web Services: Übersicht über die Sicherheitsprozesse beschrieben sind.

Sie verwenden durch AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf QLDB zuzugreifen. Kunden müssen Transport Layer Security (TLS) 1.0 oder neuer unterstützen. Wir empfehlen TLS 1.2 oder höher. Clients müssen außerdem Verschlüsselungssammlungen mit PFS (Perfect Forward Secrecy) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) unterstützen. Die meisten modernen Systemen wie Java 7 und höher unterstützen diese Modi.

Darüber hinaus müssen Anforderungen mit programmgesteuerten Anmeldeinformationen signiert werden, die einem IAM-Prinzipal zugeordnet sind. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Sie können auch einen Virtual Private Cloud (VPC)-Endpunkt für QLDB verwenden. Schnittstellen-VPC-Endpunkte ermöglichen es Ihren Amazon-VPC-Ressourcen, ihre privaten IP-Adressen für den Zugriff auf QLDB zu verwenden, ohne dem öffentlichen Internet ausgesetzt zu sein. Weitere Informationen finden Sie unter Zugriff auf Amazon QLDB über einen Schnittstellenendpunkt (AWS PrivateLink).

Zugriff auf Amazon QLDB über einen Schnittstellenendpunkt (AWS PrivateLink)

Sie können verwenden AWS PrivateLink , um eine private Verbindung zwischen Ihrer VPC und Amazon QLDB herzustellen. Sie können auf QLDB wie in Ihrer VPC zugreifen, ohne die Verwendung eines Internet-Gateways, NAT-Geräts, einer VPN-Verbindung oder einer - AWS Direct Connect Verbindung. Instances in Ihrer VPC benötigen für den Zugriff auf QLDB keine öffentlichen IP-Adressen.

Sie stellen diese private Verbindung her, indem Sie einen Schnittstellen-Endpunkt erstellen, der von AWS PrivateLinkunterstützt wird. Wir erstellen eine Endpunkt-Netzwerkschnittstelle in jedem Subnetz, das Sie für den Schnittstellen-Endpunkt aktivieren. Dies sind vom Anforderer verwaltete Netzwerkschnittstellen, die als Eintrittspunkt für Datenverkehr dienen, der für QLDB bestimmt ist.

Weitere Informationen finden Sie unter Zugriff auf AWS-Services über AWS PrivateLink im AWS PrivateLink -Leitfaden.

Themen

- Überlegungen für QLDB
- Erstellen eines Schnittstellenendpunkts für QLDB
- Erstellen einer Endpunktrichtlinie für Ihren Schnittstellen-Endpunkt

- Verfügbarkeit von Schnittstellenendpunkten für QLDB

Überlegungen für QLDB

Bevor Sie einen Schnittstellenendpunkt für QLDB einrichten, lesen Sie Überlegungen im AWS PrivateLink -Handbuch.

> **Note**
>
> QLDB unterstützt nur Aufrufe der Transaktionsdaten-API für QLDB-Sitzungen über den Schnittstellenendpunkt. Diese API enthält nur die -SendCommandOperation. Im `STANDARD` Berechtigungsmodus eines Ledgers können Sie Berechtigungen für bestimmte PartiQL-Aktionen in dieser API steuern.

Erstellen eines Schnittstellenendpunkts für QLDB

Sie können einen Schnittstellenendpunkt für QLDB entweder über die Amazon-VPC-Konsole oder die AWS Command Line Interface (AWS CLI) erstellen. Weitere Informationen finden Sie unter Erstellen eines Schnittstellenendpunkts im AWS PrivateLink -Leitfaden.

Erstellen Sie einen Schnittstellenendpunkt für QLDB mit dem folgenden Servicenamen:

```
com.amazonaws.region.qldb.session
```

Wenn Sie ein privates DNS für den Schnittstellenendpunkt aktivieren, können Sie API-Anforderungen an QLDB unter Verwendung des standardmäßigen regionalen DNS-Namens senden. Beispiel: `session.qldb.us-east-1.amazonaws.com`

Erstellen einer Endpunktrichtlinie für Ihren Schnittstellen-Endpunkt

Eine Endpunktrichtlinie ist eine IAM-Ressource, die Sie an einen Schnittstellen-Endpunkt anfügen können. Die Standard-Endpunktrichtlinie ermöglicht vollen Zugriff auf QLDB über den Schnittstellenendpunkt. Um den Zugriff auf QLDB von Ihrer VPC aus zu steuern, fügen Sie dem Schnittstellenendpunkt eine benutzerdefinierte Endpunktrichtlinie hinzu.

Eine Endpunktrichtlinie gibt die folgenden Informationen an:

- Die Prinzipale, die Aktionen ausführen können (AWS-Konten, Benutzer und Rollen).

- Aktionen, die ausgeführt werden können
- Die Ressourcen, auf denen die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter Steuern des Zugriffs auf Services mit Endpunktrichtlinien im AWS PrivateLink -Leitfaden.

Sie können das `Condition` Feld auch in einer Richtlinie verwenden, die einem Benutzer, einer Gruppe oder einer Rolle zugeordnet ist, um den Zugriff nur von einem bestimmten Schnittstellenendpunkt aus zu erlauben. Wenn sie zusammen verwendet werden, können Endpunktrichtlinien und IAM-Richtlinien den Zugriff auf bestimmte QLDB-Aktionen für bestimmte Ledger auf einen bestimmten Schnittstellenendpunkt beschränken.

Beispiel für Endpunktrichtlinien: Beschränken des Zugriffs auf einen bestimmten QLDB-Ledger

Im Folgenden finden Sie ein Beispiel für eine benutzerdefinierte Endpunktrichtlinie für QLDB. Wenn Sie diese Richtlinie an Ihren Schnittstellenendpunkt anfügen, gewährt sie Zugriff auf die -`SendCommand`Aktion und die schreibgeschützten PartiQL-Aktionen für alle Prinzipale auf der angegebenen Ledger-Ressource. In diesem Beispiel muss sich das Ledger im `STANDARD` Berechtigungsmodus befinden.

Um diese Richtlinie zu verwenden, ersetzen Sie *`us-east-1`*, *`123456789012`* und *`myExampleLedger`* im Beispiel durch Ihre eigenen Informationen.

```
{
   "Statement": [
      {
         "Sid": "QLDBSendCommandPermission",
         "Principal": "*",
         "Effect": "Allow",
         "Action": "qldb:SendCommand",
         "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
      },
      {
         "Sid": "QLDBPartiQLReadOnlyPermissions",
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "qldb:PartiQLSelect",
            "qldb:PartiQLHistoryFunction"
         ],
         "Resource": [
```

```
            "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
            "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
         ]
      }
   ]
}
```

Beispiel für eine IAM-Richtlinie: Beschränken des Zugriffs auf einen QLDB-Ledger nur von einem bestimmten Schnittstellenendpunkt aus

Im Folgenden finden Sie ein Beispiel für eine identitätsbasierte IAM-Richtlinie für QLDB. Wenn Sie diese Richtlinie an einen Benutzer, eine Rolle oder eine Gruppe anfügen, erlaubt sie den `SendCommand` Zugriff auf eine Ledger-Ressource nur vom angegebenen Schnittstellenendpunkt aus.

Um diese Richtlinie zu verwenden, ersetzen Sie *us-east-1*, *123456789012myExampleLedger*, und *vpce-1a2b3c4d* im Beispiel durch Ihre eigenen Informationen.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AccessFromSpecificInterfaceEndpoint",
         "Effect": "Deny",
         "Action": "qldb:SendCommand",
         "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger",
         "Condition": {
            "StringNotEquals": {
               "aws:sourceVpce": "vpce-1a2b3c4d"
            }
         }
      }
   ]
}
```

Verfügbarkeit von Schnittstellenendpunkten für QLDB

Amazon QLDB unterstützt Schnittstellenendpunkte mit Richtlinien in allen AWS-Regionen , in denen QLDB verfügbar ist. Eine vollständige Liste der verfügbaren Regionen finden Sie unter Amazon-QLDB-Endpunkte und -Kontingente im Allgemeine AWS-Referenz.

Fehlerbehebung für Amazon QLDB

Die folgenden Abschnitte enthalten eine zusammengefasste Liste der häufigsten Fehler, die bei der Verwendung von Amazon QLDB auftreten können, sowie Anleitungen zu deren Behebung.

Anleitungen zur Problembehandlung speziell für den IAM-Zugriff finden Sie unterFehlerbehebung für Amazon-QLDB-Identität und -Zugriff.

Bewährte Methoden zur Optimierung Ihrer PartiQL-Anweisungen finden Sie unterOptimieren der Abfrageleistung.

Themen

- Transaktionen mit dem QLDB-Treiber ausführen
- Journaldaten exportieren
- Journaldaten streamen
- Bestätigen von Journaldaten

Transaktionen mit dem QLDB-Treiber ausführen

In diesem Abschnitt werden häufig auftretende Ausnahmen aufgeführt, die der Amazon QLDB-Treiber zurückgeben kann, wenn Sie ihn zur Ausführung von PartiQL-Transaktionen in einem Ledger verwenden. Weitere Informationen über diese Funktion finden Sie unter Erste Schritte mit dem Treiber. Bewährte Methoden für die Konfiguration und Verwendung des Treibers finden Sie unterTreiberempfehlungen.

Jede Ausnahme enthält die spezifische Fehlermeldung, gefolgt von einer kurzen Beschreibung und Vorschlägen für mögliche Lösungen.

CapacityExceededException

> Meldung: Kapazität überschritten
>
> Amazon QLDB hat die Anfrage abgelehnt, weil sie die Verarbeitungskapazität des Ledgers überschritten hat. QLDB setzt ein internes Skalierungslimit pro Ledger durch, um den Zustand und die Leistung des Dienstes aufrechtzuerhalten. Dieses Limit variiert je nach Workload-Größe jeder einzelnen Anfrage. Beispielsweise kann eine Anforderung eine erhöhte Arbeitslast haben,

wenn sie ineffiziente Datentransaktionen durchführt, z. B. Tabellenscans, die aus einer nicht indexqualifizierten Abfrage resultieren.

Wir empfehlen, dass Sie warten, bevor Sie die Anfrage erneut versuchen. Wenn Ihre Anwendung immer wieder auf diese Ausnahme stößt, optimieren Sie Ihre Kontoauszüge und reduzieren Sie die Anzahl und das Volumen der Anfragen, die Sie an das Hauptbuch senden. Beispiele für die Optimierung von Kontoauszügen sind die Ausführung von weniger Anweisungen pro Transaktion und die Optimierung Ihrer Tabellenindizes. Informationen zur Optimierung von Anweisungen und zur Vermeidung von Tabellenscans finden Sie unterOptimieren der Abfrageleistung.

Wir empfehlen außerdem die Verwendung QLDB. Der Treiber verfügt über eine Standardrichtlinie für Wiederholungen, die Exponential Backoff und Jitter verwendet, um bei solchen Ausnahmen automatisch den Vorgang zu wiederholen. Das Konzept des exponentiellen Backoffs besteht darin, für aufeinanderfolgende Fehlerantworten zunehmend längere Wartezeiten zwischen Wiederholungen zu verwenden.

InvalidSessionException

Meldung: Transaction *`transactionId`* has expired (Transaktion abgelaufen)

Eine Transaktion hat ihre maximale Lebensdauer überschritten. Eine Transaktion kann bis zu 30 Sekunden lang ausgeführt werden, bevor sie bestätigt wird. Nach diesem Timeout-Limit werden alle an der Transaktion ausgeführten Arbeiten zurückgewiesen, und QLDB verwirft die Sitzung. Dieses Limit schützt den Kunden vor undichten Sitzungen, indem Transaktionen gestartet werden und diese nicht abgeschlossen oder storniert werden.

Wenn dies eine häufige Ausnahme in Ihrer Anwendung ist, dauert es wahrscheinlich, dass die Ausführung von Transaktionen einfach zu lange dauert. Wenn die Transaktionslaufzeit länger als 30 Sekunden dauert, optimieren Sie Ihre Kontoauszüge, um die Transaktionen zu beschleunigen. Beispiele für die Optimierung von Kontoauszügen sind die Ausführung von weniger Anweisungen pro Transaktion und die Optimierung Ihrer Tabellenindizes. Weitere Informationen finden Sie unter Optimieren der Abfrageleistung.

InvalidSessionException

Meldung: Session *`sessionId`* has expired (Sitzung ist abgelaufen)

QLDB hat die Sitzung verworfen, weil sie ihre maximale Gesamtlebensdauer überschritten hat. QLDB verwirft Sitzungen nach 13—17 Minuten, unabhängig von einer aktiven Transaktion. Sitzungen können aus einer Reihe von Gründen verloren gehen oder beeinträchtigt werden, wie z. B. Hardwarefehler, Netzwerkfehler oder Anwendungsneustarts. Daher erzwingt QLDB

eine maximale Lebensdauer von Sitzungen, um sicherzustellen, dass die Clientsoftware gegen Sitzungsausfälle resistent ist.

Wenn diese Ausnahme auftritt, empfehlen wir, eine neue Sitzung abzurufen und die Transaktion erneut zu versuchen. Wir empfehlen außerdem, die neueste Version des QLDB-Treibers zu verwenden, der den Sitzungspool und seinen Zustand im Namen der Anwendung verwaltet.

InvalidSessionException

Meldung: No such session (Keine solche Sitzung)

Der Client hat versucht, mit QLDB über eine Sitzung zu handeln, die nicht existiert. Unter der Annahme, dass der Client eine Sitzung verwendet, die zuvor existiert hat, ist die Sitzung möglicherweise aus einem der folgenden Gründe nicht mehr vorhanden:

- Wenn in einer Sitzung ein interner Serverausfall auftritt (d. h. ein Fehler mit dem HTTP-Antwortcode 500), entscheidet sich QLDB möglicherweise dafür, die Sitzung vollständig zu verwerfen, anstatt dem Kunden zu erlauben, Transaktionen mit einer Sitzung mit unsicherem Status durchzuführen. Dann schlagen alle Wiederholungsversuche in dieser Sitzung mit diesem Fehler fehl.
- Abgelaufene Sitzungen werden irgendwann von QLDB vergessen. Dann führen alle Versuche, die Sitzung weiter zu verwenden, zu diesem Fehler anstatt des ursprünglichen `InvalidSessionException`.

Wenn diese Ausnahme auftritt, empfehlen wir, eine neue Sitzung abzurufen und die Transaktion erneut zu versuchen. Wir empfehlen außerdem, die neueste Version des QLDB-Treibers zu verwenden, der den Sitzungspool und seinen Zustand im Namen der Anwendung verwaltet.

RateExceededException

Meldung: The rate was exceeded (Die Rate wurde überschritten)

QLDB drosselte einen Client auf der Grundlage der Identität des Anrufers. QLDB erzwingt die Drosselung pro Region und pro Konto mithilfe eines Token-Bucket-Drosselungsalgorithmus. QLDB tut dies, um die Leistung des Dienstes zu unterstützen und um eine faire Nutzung für alle QLDB-Kunden sicherzustellen. Beispielsweise kann der Versuch, eine große Anzahl gleichzeitiger Sitzungen mit dem `StartSessionRequest`-Vorgang abzurufen, zu einer Drosselung führen.

Um den korrekten Anwendungszustand aufrechtzuerhalten und eine weitere Drosselung zu verringern, können Sie diese Ausnahme mit exponentiellem Backoff und Jitter wiederholen. Das Konzept des exponentiellen Backoffs besteht darin, für aufeinanderfolgende Fehlerantworten zunehmend längere Wartezeiten zwischen Wiederholungen zu verwenden. Wir empfehlen

die Verwendung. Wir QLDB Verwendung. Der Treiber verfügt über eine Standardrichtlinie für Wiederholungen, die exponentielles Backoff und Jitter verwendet, um es bei Ausnahmen wie diesen automatisch zu wiederholen.

Die neueste Version des QLDB-Treibers kann auch hilfreich sein, wenn Ihre Anwendung ständig von QLDB für`StartSessionRequest` Anrufe gedrosselt wird. Der Treiber verwaltet einen Pool von Sitzungen, die transaktionsübergreifend wiederverwendet werden. Dies kann dazu beitragen, die Anzahl der`StartSessionRequest` Aufrufe zu reduzieren, die Ihre Anwendung tätigt. Um eine Erhöhung der API-Drosselungslimits anzufordern, wenden Sie sich an das AWS SupportCenter.

LimitExceededException

Meldung: Exceeded the session limit (Sitzungslimit überschritten)

Ein Ledger hat sein Kontingent (auch als Limit bezeichnet) für die Anzahl aktiver Sitzungen überschritten. Dieses Kontingent ist in Kontingente und Limits in Amazon QLDB definiert. Die Anzahl aktiver Sitzungen eines Ledgers ist schließlich konsistent, und Ledger, die konsistent in der Nähe des Kontingents ausgeführt werden, erfahren diese Ausnahme möglicherweise in regelmäßigen Abständen.

Um die Integrität Ihrer Anwendung aufrechtzuerhalten, empfehlen wir, bei dieser Ausnahme einen erneuten Versuch zu machen. Um diese Ausnahme zu vermeiden, stellen Sie sicher, dass Sie nicht mehr als 1.500 gleichzeitige Sitzungen für ein einzelnes Ledger über alle Clients hinweg konfiguriert haben. Sie können beispielsweise die maxConcurrentTransactionsMethode des Amazon QLDB-Treibers für Java verwenden, um die maximale Anzahl verfügbarer Sitzungen in einer Treiberinstanz zu konfigurieren.

QldbClientException

Meldung: A streamed result is only valid when the parent transaction is open (Ein gestreamtes Ergebnis ist nur gültig, wenn die übergeordnete Transaktion geöffnet ist)

Die Transaktion ist abgeschlossen und kann nicht verwendet werden, um die Ergebnisse von QLDB abzurufen. Eine Transaktion wird geschlossen, wenn sie entweder bestätigt oder storniert wurde.

Diese Ausnahme tritt auf, wenn der Client direkt mit dem`Transaction` Objekt arbeitet und versucht, Ergebnisse von QLDB abzurufen, nachdem er eine Transaktion bestätigt oder storniert hat. Um dieses Problem zu beheben, muss der Kunde die Daten lesen, bevor er die Transaktion abschließt.

Journaldaten exportieren

In diesem Abschnitt werden häufig auftretende Ausnahmen aufgeführt, die QLDB zurückgeben kann, wenn Sie Journaldaten aus einem Ledger in einen Amazon S3 S3-Bucket exportieren. Weitere Informationen über diese Funktion finden Sie unter Exportieren von Journaldaten aus Amazon QLDB.

Jede Ausnahme enthält die spezifische Fehlermeldung, gefolgt von einer kurzen Beschreibung und Vorschlägen für mögliche Lösungen.

AccessDeniedException

Nachricht: Benutzer: *`UserArn`* ist nicht berechtigt, Folgendes auszuführen: iam:PassRole on resource: *`roleARN`*

Sie sind nicht berechtigt, eine IAM-Rolle an den QLDB Service zu übergeben. QLDB benötigt eine Rolle für alle Journal-Exportanfragen, und Sie müssen über die erforderlichen Berechtigungen verfügen, um diese Rolle an QLDB zu übergeben. Die Rolle gewährt QLDB Schreibberechtigungen in Ihrem angegebenen Amazon S3 S3-Bucket.

Stellen Sie sicher, dass Sie eine IAM-Richtlinie definieren, die die Berechtigung zur Ausführung des`PassRole` API-Vorgangs für Ihre angegebene IAM-Rollenressource für den QLDB-Dienst (`qldb.amazonaws.com`) gewährt. Eine Beispielrichtlinie finden Sie in Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

IllegalArgumentException

Meldung: QLDB encountered an error validating S3 configuration: *`errorCode errorMessage`*

Eine mögliche Ursache für diesen Fehler ist, dass der bereitgestellte in Amazon S3 nicht vorhanden ist. Oder QLDB hat nicht genügend Berechtigungen, um Objekte in Ihren angegebenen Amazon S3 S3-Bucket zu schreiben.

Stellen Sie sicher, dass der S3-Bucket-Name, den Sie in Ihrer Anforderung für den Exportauftrag angeben, korrekt ist. Weitere Informationen zur Bucket-Benennung finden Sie unter Bucket-Einschränkungen und -Einschränkungen im Amazon Simple Storage Service User Guide.

Stellen Sie außerdem sicher, dass Sie eine Richtlinie für Ihren angegebenen Bucket definieren, die den QLDB-Dienst (`qldb.amazonaws.com`) gewährt`PutObject` und ihm`PutObjectAcl` Berechtigungen gewährt. Weitere Informationen hierzu finden Sie unter Exportberechtigungen.

IllegalArgumentException

Meldung: Unerwartete Antwort von Amazon S3 bei der Validierung der S3-Konfiguration. Antwort von S3: *errorCode ErrorMessage*

Der Versuch, Journalexportdaten in den bereitgestellten S3-Bucket zu schreiben, schlug mit der bereitgestellten Amazon S3 S3-Fehlerantwort fehl. Weitere Informationen zu möglichen finden Sie unter Fehlerbehebung für Amazon S3 im Amazon Simple Storage Service User Guide.

IllegalArgumentException

Meldung: Amazon S3 bucket prefix must not exceed 128 characters

Das in der Journal-Journalexportanfrage angegebene Präfix enthält mehr als 128 Zeichen.

IllegalArgumentException

Meldung: Start date must not be greater than end date

Sowohl `InclusiveStartTime` als auch `ExclusiveEndTime` müssen im Datums- und Zeitformat ISO 8601 und in koordinierter Weltzeit (Coordinated Universal Time, UTC) angegeben werden.

IllegalArgumentException

Meldung: End date cannot be in the future

Sowohl `InclusiveStartTime` als auch `ExclusiveEndTime` müssen im Datums- und Zeitformat `ISO 8601` und in koordinierter Weltzeit (Coordinated Universal Time, UTC) angegeben werden.

IllegalArgumentException

Meldung: Die angegebene Objektverschlüsselungseinstellung (S3EncryptionConfiguration) ist nicht mit einemAWS Key Management Service (AWS KMS) -Schlüssel kompatibel

Sie haben einen `KMSKeyArn` mit einem `ObjectEncryptionType` vom entweder `NO_ENCRYPTION` oder `SSE_S3` angegeben. Sie können nur einen Kunden angeben, derAWS KMS key für einen Objektverschlüsselungstyp von verwaltet wird`SSE_KMS`. Weitere Informationen zu serverseitigen Verschlüsselung in Amazon S3 finden Sie unter Schützen von Daten mithilfe serverseitiger Verschlüsselung im Amazon S3 S3-Entwicklerhandbuch.

LimitExceededException

Meldung: Exceeded the limit of 2 concurrently running Journal export jobs

QLDB erzwingt eine Standardbeschränkung von zwei gleichzeitigen Journalexportaufträgen.

Journaldaten streamen

In diesem Abschnitt werden häufig auftretende Ausnahmen aufgeführt, die QLDB zurückgeben kann, wenn Sie Journaldaten aus einem Ledger an Amazon Kinesis Data Streams streamen. Weitere Informationen über diese Funktion finden Sie unter Streaming von Journaldaten aus Amazon QLDB.

Jede Ausnahme enthält die spezifische Fehlermeldung, gefolgt von einer kurzen Beschreibung und Vorschlägen für mögliche Lösungen.

AccessDeniedException

Nachricht: Benutzer: *`UserArn`* ist nicht berechtigt, Folgendes auszuführen: iam:PassRole on resource: *`roleARN`*

Sie sind nicht berechtigt, eine IAM-Rolle an den QLDB Service zu übergeben. QLDB benötigt eine Rolle für alle Journal-Stream-Anfragen, und Sie müssen über die erforderlichen Berechtigungen verfügen, um diese Rolle an QLDB zu übergeben. Die Rolle gewährt QLDB Schreibberechtigungen für Ihre angegebene Amazon Kinesis Data Streams Streams-Ressource.

Stellen Sie sicher, dass Sie eine IAM-Richtlinie definieren, die die Berechtigung zur Ausführung des`PassRole` API-Vorgangs für Ihre angegebene IAM-Rollenressource für den QLDB-Dienst (`qldb.amazonaws.com`) gewährt. Eine Beispielrichtlinie finden Sie in Beispiele für identitätsbasierte Richtlinien für Amazon QLDB.

IllegalArgumentException

Meldung: QLDB hat einen Fehler beim Validieren von Kinesis Data Streams festgestellt: Antwort von Kinesis: *`errorCode errorMessage`*

Eine mögliche Ursache für diesen Fehler ist, dass die bereitgestellte Kinesis Data Streams Streams-Ressource nicht existiert. Oder QLDB hat nicht genügend Berechtigungen, um Datensätze in Ihren angegebenen Kinesis-Datenstream zu schreiben.

Stellen Sie sicher, dass der Kinesis-Datenstream, den Sie in Ihrer Stream-Anfrage angeben, korrekt ist. Weitere Informationen finden Sie unter Erstellen und Aktualisieren von Datenstreams im Amazon-Kinesis-Data-Streams-Entwicklerhandbuch.

Stellen Sie außerdem sicher, dass Sie eine Richtlinie für Ihren angegebenen Kinesis-Datenstream definieren, die dem QLDB-Dienst (`qldb.amazonaws.com`) Berechtigungen für die folgenden Aktionen gewährt. Weitere Informationen finden Sie unter Stream-Berechtigungen.

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

Meldung: Unerwartete Antwort von Kinesis Data Streams bei der Validierung der Kinesis-Konfiguration. Antwort von Kinesis: *`errorCode ErrorMessage`*

Der Versuch, Datensätze in den bereitgestellten Kinesis-Datenstrom zu schreiben, schlug mit der bereitgestellten Kinesis-Fehlerantwort fehl. Weitere Informationen zu möglichen finden Sie unter Fehlerbehebung für Amazon Kinesis Data Streams Streams-Produzenten im Amazon-Kinesis-Data-Streams-Entwicklerhandbuch.

IllegalArgumentException

Meldung: Start date must not be greater than end date.

Sowohl `InclusiveStartTime` als auch `ExclusiveEndTime` müssen im Datums- und Zeitformat ISO 8601 und in koordinierter Weltzeit (Coordinated Universal Time, UTC) angegeben werden.

IllegalArgumentException

Meldung: Startdatum kann nicht in der Zukunft liegen.

Sowohl `InclusiveStartTime` als auch `ExclusiveEndTime` müssen im Datums- und Zeitformat `ISO 8601` und in koordinierter Weltzeit (Coordinated Universal Time, UTC) angegeben werden.

LimitExceededException

Meldung: Grenzwert von 5 parallel ausgeführten Journal-Streams zu Kinesis Data Streams überschritten

QLDB erzwingt eine Standardbeschränkung von fünf gleichzeitigen Journal-Streams.

Bestätigen von Journaldaten

In diesem Abschnitt werden häufig vorkommende Ausnahmen aufgeführt, die QLDB zurückgeben kann, wenn Sie Journaldaten in einem Ledger überprüfen. Weitere Informationen über diese Funktion finden Sie unter Datenverifizierung in Amazon QLDB.

Jede Ausnahme enthält die spezifische Fehlermeldung, gefolgt von den API-Vorgängen, die sie auslösen können, einer kurzen Beschreibung und Vorschlägen für mögliche Lösungen.

IllegalArgumentException

Nachricht: Der angegebene Ion-Wert ist ungültig und kann nicht analysiert werden.

API-Operationen:`GetDigest, GetBlock, GetRevision`

Stellen Sie sicher, dass Sie einen gültigen Amazon Ion-Wert bereitstellen, bevor Sie Ihre Anfrage erneut stellen.

IllegalArgumentException

Meldung: Die angegebene Blockadresse ist ungültig.

API-Operationen:`GetDigest, GetBlock, GetRevision`

Stellen Sie sicher, dass Sie eine gültige Block-Adresse bereitstellen, bevor Sie Ihre Anfrage erneut stellen. Ein Block-Adresse ist eine Amazon Ion-Struktur, die über zwei Felder verfügt: `strandId` und `sequenceNo`.

Beispiel: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

IllegalArgumentException

Meldung: Die Sequenznummer der bereitgestellten Digest-Tip-Adresse liegt außerhalb des letzten Commiting-Datensatzes des Strangs.

API-Operationen:`GetDigest, GetBlock, GetRevision`

Die Digest-Tip-Adresse, die Sie angeben, muss über eine Sequenznummer verfügen, die kleiner oder gleich der Sequenznummer des zuletzt übertragenen Datensatzes der Journal-Strähne ist. Bevor Sie Ihre Anforderung wiederholen, stellen Sie sicher, dass Sie eine Digest-Tip-Adresse mit einer gültigen Sequenznummer bereitstellen.

IllegalArgumentException

Meldung: The Strand ID of the provided block address is not valid.

API-Operationen:`GetDigest, GetBlock, GetRevision`

Die von Ihnen angegebene Block-Adresse muss eine Strähnen-ID aufweisen, die der Strähnen-ID des Journals entspricht. Bevor Sie Ihre Anforderung wiederholen, stellen Sie sicher, dass Sie eine Block-Adresse mit einer gültigen Strähnen-ID bereitstellen.

IllegalArgumentException

Meldung: Die Sequenznummer der bereitgestellten Blockadresse liegt außerhalb des zuletzt eingegebenen Datensatzes des Strangs.

API-Operationen:`GetBlock, GetRevision`

Die Block-Adresse, die Sie angeben, muss über eine Sequenznummer verfügen, die kleiner oder gleich der Sequenznummer des zuletzt übertragenen Datensatzes der Strähne ist. Bevor Sie Ihre Anforderung wiederholen, stellen Sie sicher, dass Sie eine Block-Adresse mit einer gültigen Sequenznummer.

IllegalArgumentException

Nachricht: Die Strähnen-ID der bereitgestellten Block-Adresse muss der Strähnen-ID der bereitgestellten Digest-Tip-Adresse entsprechen.

API-Operationen:`GetBlock, GetRevision`

Sie können ein Dokument nur überprüfen oder blockieren, wenn es in derselben Journal-Strähne vorhanden ist wie das von Ihnen angegebene Digest-Journal.

IllegalArgumentException

Nachricht: Die Sequenznummer der bereitgestellten Block-Adresse darf nicht größer sein als die Sequenznummer der bereitgestellten Digest-Tip-Adresse.

API-Operationen:`GetBlock, GetRevision`

Sie können eine Dokumentrevision nur überprüfen oder blockieren, wenn sie von dem von Ihnen bereitgestellten Digest abgedeckt wird. Dies bedeutet, dass sie vor der Digest-Tip-Adresse an das Journal übertragen wurde.

IllegalArgumentException

Nachricht: Die angegebene Dokument-ID wurde im Block an der angegebenen Block-Adresse nicht gefunden.

API-Betrieb:`GetRevision`

Die Dokument-ID, die Sie angeben, muss in der von Ihnen angegebenen Block-Adresse vorhanden sein. Bevor Sie Ihre Anforderung wiederholen, stellen Sie sicher, dass diese beiden Parameter konsistent sind.

Amazon QLDB PartiQL-Referenz

Amazon QLDB unterstützt eine Teilmenge der PartiQL-Abfragesprache. In den folgenden Themen wird die QLDB Implementierung von PartiQL beschrieben.

> **Note**
>
> - QLDB unterstützt nicht alle PartiQL-Operationen.
> - Alle PartiQL-Anweisungen in QLDB unterliegen Transaktionslimits, wie in definiertKontingente und Limits in Amazon QLDB.
> - Diese Referenz enthält grundlegende Syntax- und Verwendungsbeispiele für PartiQL-Anweisungen, die Sie manuell auf der QLDB-Konsole oder der QLDB-Shell ausführen. Codebeispiele, die zeigen, wie ähnliche Anweisungen mithilfe des QLDB-Treibers programmgesteuert ausgeführt werden, finden Sie in den Tutorials unterErste Schritte mit dem Treiber.

Themen

Was ist PartiQL?

PartiQL bietet SQL-kompatiblen Abfragezugriff über mehrere Datenspeicher mit strukturierten, halbstrukturierten und verschachtelten Daten hinweg. Es wird in Amazon häufig verwendet und ist jetzt als Teil vieler verfügbarAWS-Services, einschließlich QLDB.

Die PartiQL-Spezifikation und ein Tutorial zur Core-Abfragesprache finden Sie in der PartiQL-Dokumentation.

PartiQL erweitert SQL-92, um Dokumente im Amazon Ion-Datenformat zu unterstützen. Weitere Informationen über Amazon Ion finden Sie unter Referenz zum Amazon Ion-Datenformat in Amazon QLDB.

PartiQL in Amazon QLDB

Zum Ausführen von PartiQL-Abfragen in QLDB können Sie eine der folgenden Optionen verwenden:

- Der PartiQL-Editor auf derAWS Management Console für QLDB
- Die Kommandozeilen-QLDB-Shell
- EinAWS bereitgestellter QLDB-Treiber zum programmgesteuerten Ausführen von Abfragen

Weitere Informationen zum Verwenden dieser Methoden für den Zugriff auf QLDB finden Sie unterZugreifen auf Amazon QLDB.

Informationen dazu, wie Sie den Zugriff steuern, um jeden PartiQL-Befehl für bestimmte Tabellen auszuführen, finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

PartiQL Kurztipps in QLDB

Im Folgenden finden Sie eine kurze Zusammenfassung der Tipps und bewährten Methoden für die Arbeit mit PartiQL in QLDB:

- Machen Sie sich mit Parallelität und Transaktionslimits vertraut — Alle Anweisungen, einschließlich`SELECT` Abfragen, unterliegen Optimist Concurrency Control (OCC) -Konflikten und Transaktionslimits, einschließlich eines Transaktions-Timeouts von 30 Sekunden.

- Indizes verwenden — Verwenden Sie Indizes mit hoher Kardinalität und führen Sie gezielte Abfragen aus, um Ihre Aussagen zu optimieren und vollständige Tabellenscans zu vermeiden. Weitere Informationen hierzu finden Sie unter Optimieren der Abfrageleistung.
- Verwenden Sie Gleichheitsprädikate — Indexierte Suchvorgänge erfordern einen Gleichheitsoperator (`=` oder `IN`). Ungleichheitsoperatoren (`<>` `LIKE`, , `BETWEEN`) eignen sich nicht für indexierte Suchvorgänge und führen zu vollständigen Tabellenscans.
- Nur innere Joins verwenden — QLDB unterstützt nur innere Joins. Es hat sich bewährt, Felder miteinander zu verknüpfen, die für jede Tabelle, die Sie verknüpfen, indexiert sind. Wählen Sie Indizes mit hoher Kardinalität sowohl für die Verbindungskriterien als auch für die Gleichheitsprädikate.

Amazon QLDB PartiQL-Referenzkonventionen

In diesem Abschnitt werden die Konventionen erklärt, die für die Beschreibung der PartiQL-Befehle, -Funktionen und -Ausdrücke in der Amazon-QLDB-PartiQL-Referenz verwendet werden. Diese Konventionen dürfen nicht mit der Syntax und Semantik der PartiQL-Abfragesprache selbst verwechselt werden.

Zeichen	Beschreibung
GROSSBUCHSTABEN	Wörter in Großbuchstaben sind Schlüsselwörter.
[ ]	Eckige Klammern bezeichnen optionale Argumente oder Klauseln. Mehrere Argumente in eckigen Klammern zeigen an, dass Sie eine beliebige Anzahl der Argumente verwenden können. Argumente in eckigen Klammern, die jeweils in einer eigenen Zeile stehen, zeigen außerdem an, dass der QLDB-Parser die Argumente in der Reihenfolge erwartet, in der sie in der Syntax aufgelistet sind.
|	Pipe-Zeichen zeigen an, dass Sie zwischen den Argumenten wählen können.
Rote Kursivschrift	Wörter in roter Kursivschrift zeigen Platzhalter an. Sie müssen das kursiv formatierte rote Wort durch den entsprechenden Wert ersetzen.

Zeichen	Beschreibung
. . .	Auslassungspunkte zeigen an, dass Sie das Element davor wiederholen können.
'	Werte in einfachen Anführungszeichen müssen zusammen mit den einfachen Anführungszeichen verwendet werden. In PartiQL bezeichnen einfache Anführungszeichen Zeichenfolgenwerte oder Feldnamen in Amazon Ion-Strukturen.
"	Werte in doppelten Anführungszeichen müssen zusammen mit den doppelten Anführungszeichen verwendet werden. In PartiQL bezeichnen doppelte Anführungszeichen Bezeichner in Anführungszeichen.
`	Werte in Backticks geben an, dass Sie die Backticks eingeben müssen. In PartiQL stehen Backticks für Ion-Literalwerte.

Datentypen in Amazon QLDB

Amazon QLDB speichert Dokumente im Amazon Ion-Format. Amazon Ion ist ein Datenserialisierungsformat (sowohl in Textform als auch in Binär-kodierter Form), das eine Obermenge von JSON ist. In der folgenden Tabelle sind Ion-Datentypen aufgeführt, die Sie in QLDB-Dokumenten verwenden können.

Datentyp	Beschreibung
`null`	Ein generischer Nullwert
`bool`	Boolesche Werte
`int`	Signierte Ganzzahlen von beliebiger Größe.
`decimal`	Dezimal-kodierte echte Zahlen von beliebiger Präzision
`float`	Binär-kodierte Gleitkommazahlen (64-Bit-IEEE)

Datentyp	Beschreibung
`timestamp`	Datums-/Uhrzeit-/Zeitzonen-Momente von beliebiger Präzision
`string`	Unicode-Textliterale
`symbol`	Unicode-symbolische Atome (d. h. Kennungen)
`blob`	Binäre Daten der benutzerdefinierten Kodierung
`clob`	Textdaten der benutzerdefinierten Kodierung
`struct`	Nicht geordnete Sammlungen von Schlüssel-Wert-Paaren
`list`	Zulässige heterogene Sammlungen von Werten

Eine vollständige Liste der Ion-Kerndatentypen mit vollständigen Beschreibungen und Details zur Wertformatierung finden Sie im Ion-Spezifikationsdokument auf der GitHub Amazon-Website.

Amazon QLDB Dokumente

Amazon QLDB speichert Datensätze als Dokumente, bei denen es sich lediglich um Amazon`struct` Ion-Objekte handelt, die in eine Tabelle eingefügt werden. Die Ion-Spezifikation finden Sie auf der Amazon GitHub Ion-Website.

Themen

- Ion-Dokumentstruktur
- Partielle Ionentypzuordnung
- Dokument-ID

Ion-Dokumentstruktur

Wie JSON bestehen QLDB-Dokumente aus Name-Wert-Paaren in der folgenden Struktur.

```
{
```

```
  name1: value1,
  name2: value2,
  name3: value3,
  ...
  nameN: valueN
}
```

Die Namen sind Symbol-Token, und die Werte sind nicht eingeschränkt. Jedes Name-Wert-Paar wird als Feld bezeichnet. Der Wert eines Felds kann einer der Ion-Datentypen sein, einschließlich Containertypen: verschachtelte Strukturen, Listen und Listen von Strukturen.

Ebenso wie JSON`struct` wird a durch geschweifte Klammern (`{...}`) und `alist` durch eckige Klammern (`[...]`) bezeichnet. Das folgende Beispiel ist ein Dokument aus den Beispieldaten in Erste Schritte mit der Amazon QLDB-Konsole, das Werte verschiedener Typen enthält.

```
{
    VIN: "1N4AL11D75C109151",
    LicensePlateNumber: "LEWISR261LL",
    State: "WA",
    City: "Seattle",
    PendingPenaltyTicketAmount: 90.25,
    ValidFrom: 2017-08-21T,
    ValidTo: 2020-05-11T,
    Owners: {
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    }
}
```

> **Important**
>
> In Ion stehen doppelte Anführungszeichen für Zeichenfolgenwerte und Symbole ohne Anführungszeichen für Feldnamen. In PartiQL bezeichnen einfache Anführungszeichen jedoch sowohl Zeichenfolgen als auch Feldnamen.
> Dieser Unterschied in der Syntax ermöglicht die PartiQL-Abfragesprache zur Wahrung der SQL-Kompatibilität und das Amazon Ion-Datenformat zur Wahrung der JSON-Kompatibilität. Einzelheiten zur Syntax und Semantik von PartiQL in QLDB finden Sie unterAbfragen von Ion mit PartiQL.

Partielle Ionentypzuordnung

In QLDB erweitert PartiQL das Typsystem von SQL, um das Ion-Datenmodell abzudecken. Diese Zuordnung wird nachfolgend beschrieben:

- Skalare SQL-Typen sind von Ihren Ion-Pendants abgedeckt. Beispiel:
  - `CHAR` und `VARCHAR` sind Unicode-Sequenzen, die dem Ion-`string`-Typ zugeordnet werden.
  - `NUMBER` entspricht dem Ion-`decimal`-Typ.
- Der`struct` Typ von Ion entspricht einem SQL-Tupel, das traditionell eine Tabellenzeile darstellt.
  - Bei offenem Inhalt und ohne Schema werden Abfragen, die auf der geordneten Natur eines SQL-Tupels beruhen, jedoch nicht unterstützt (z. B. die Ausgabereihenfolge von`SELECT *`).
- Zusätzlich zu `NULL` hat PartiQL einen `MISSING`-Typ. Hierbei handelt es sich um eine Spezialisierung von `NULL`, die das Fehlen eines Feldes angibt. Dieser Typ ist notwendig, da Ion `struct`-Felder möglicherweise knapp sind.

Dokument-ID

QLDB weist jedem Dokument, das Sie in eine Tabelle einfügen, eine Dokument-ID zu. Alle vom System zugewiesenen IDs sind universell eindeutige Identifikatoren (UUID), die jeweils in einer Base62-kodierten Zeichenfolge dargestellt werden (z. B.`3Qv67yjXEwB9SjmvkuG6Cp`). Weitere Informationen finden Sie unter Eindeutige IDs in Amazon QLDB.

Jede Dokumentrevision wird eindeutig durch eine Kombination aus Dokument-ID und nullbasierter Versionsnummer identifiziert.

Diese Dokument-ID- und Versionsfelder sind in den Metadaten des Dokuments enthalten, die Sie in der festgeschriebenen Ansicht (d. h. in der vom System definierten Ansicht der Tabelle) abrufen können. Weitere Hinweise zu Ansichten in QLDB finden Sie unterSchlüsselkonzepte. Weitere Informationen zu Metadaten finden Sie unter Metadaten von Dokumenten abfragen.

Ion mit PartiQL in Amazon QLDB abfragen

Wenn Sie Daten in Amazon QLDB abfragen, schreiben Sie Anweisungen im PartiQL-Format, aber QLDB gibt Ergebnisse im Amazon Ion-Format zurück. PartiQL soll SQL-kompatibel sein. Ion ist eine Erweiterung von JSON. Dies führt zu syntaktischen Unterschieden zwischen der Art und Weise, wie Sie Daten in Ihren Abfrageanweisungen notieren, und der Darstellung Ihrer Abfrageergebnisse.

In diesem Abschnitt werden die grundlegende Syntax und Semantik für die manuelle Ausführung von PartiQL-Anweisungen mithilfe der QLDB-Konsole oder der QLDB-Shell beschrieben.

> **Tip**
>
> Bei der programmgesteuerten Ausführung von PartiQL-Abfragen hat sich die Verwendung parametrisierte Anweisungen als bewährte Methode erwiesen. Sie können in Ihren Anweisungen ein Fragezeichen (?) als Platzhalter für Bind-Variablen verwenden, um diese Syntaxregeln zu vermeiden. Das ist auch sicherer und effizienter.
> Weitere Informationen finden Sie in den folgenden Tutorials unter Erste Schritte mit dem Treiber:
>
> - Java:ErstErstErstErstErste-Erst |Kochbuchreferenz
> - .NET:Erste-Schrit-T |Kochbuchreferenz
> - Gehe:Schnellstart-Tutorial |Kochbuchreferenz
> - Node.js:Erste-Schrit-T-T-TSchrit |Kochbuchreferenz — Referenz für
> - Python:Erste Schritte |Kochbuchreferenz

Themen

Syntax und Semantik

Wenn Sie die QLDB-Konsole oder die QLDB-Shell zum Abfragen von Ion-Daten verwenden, sind die grundlegenden Syntax und Semantik von PartiQL wie folgt aufgeführt:

Groß-/Kleinschreibung

Bei allen QLDB-Systemobjektnamen — einschließlich Feldnamen, Tabellennamen und Buchnamen — wird zwischen Groß- und Kleinschreibung unterschieden.

Zeichenfolgenwerte

In Ion bezeichnen doppelte Anführungszeichen (`"..."`) eine Zeichenfolge.

In PartiQL bezeichnen einfache Anführungszeichen (`'...'`) eine Zeichenfolge.

Symbole und Bezeichner

In Ion bezeichnen einfache Anführungszeichen (`'...'`) ein Symbol. Eine Teilmenge von Symbolen in Ion, die als Identifikatoren bezeichnet werden, wird durch Text ohne Anführungszeichen dargestellt.

In PartiQL bezeichnen doppelte Anführungszeichen (`"..."`) einen PartiQL-Bezeichner in Anführungszeichen, z. B. ein reserviertes Wort, das als Tabellenname verwendet wird. Text ohne Anführungszeichen steht für einen regulären PartiQL-Bezeichner, z. B. einen Tabellennamen, der kein reserviertes Wort ist.

Ion-Literale

Alle Ion-Literale können mit umgekehrten Anführungszeichen (`` `...` ``) in einer PartiQL-Anweisung bezeichnet werden.

Feldnamen

Bei Ion-Feldnamen wird die Groß-/Kleinschreibung beachtet. Mit PartiQL können Sie Feldnamen in einer DML-Anweisung mit einfachen Anführungszeichen bezeichnen. Dies ist eine Kurzbezeichnung zur Verwendung der `cast`-Funktion von PartiQL zur Definition eines Symbols. Es ist auch intuitiver als die Verwendung von Backticks, um ein wörtliches Ionensymbol zu bezeichnen.

Literale

Literale der PartiQL-Abfragesprache entsprechen den Ion-Datentypen wie folgt:

Skalare

Befolgen Sie gegebenenfalls die SQL-Syntax, wie im Abschnitt Partielle Ionentypzuordnung beschrieben. Beispiel:

- `5`
- `'foo'`

- `null`

Structs

Auch bekannt als „Tupel“ oder Objekte in vielen Formaten und anderen Datenmodellen.

Bezeichnet durch geschweifte Klammern (`{...}`) mit `struct`-Elementen, die durch Kommata getrennt sind.

- `{ 'id' : 3, 'arr': [1, 2] }`

Listen

Auch als „Arrays“ bezeichnet.

Bezeichnet durch eckige Klammern (`[...]`) mit durch Kommata getrennten Listenelementen.

- `[ 1, 'foo' ]`

Taschen

Ungeordnete Sammlungen in PartiQL.

Bezeichnet durch doppelte eckige Klammern (`<<...>>`) mit durch Kommata getrennten Taschenelementen. In QLDB kann man sich einen Tisch als Tasche vorstellen. Eine Tasche kann jedoch nicht in Dokumente in einer Tabelle verschachtelt werden.

- `<< 1, 'foo' >>`

Beispiel

Hier sehen Sie ein Beispiel für die Verwendung der Syntax für eine `INSERT`-Anweisung mit verschiedenen Ion-Typen.

```
INSERT INTO VehicleRegistration VALUE
{
    'VIN' : 'KM8SRDHF6EU074761', --string
    'RegNum' : 1722, --integer
    'State' : 'WA',
    'City' : 'Kent',
    'PendingPenaltyTicketAmount' : 130.75, --decimal
    'Owners' : { --nested struct
        'PrimaryOwner' : { 'PersonId': '294jJ3YUoH1IEEm8GSabOs' },
        'SecondaryOwners' : [ --list of structs
```

```
            { 'PersonId' : '1nmeDdLo3AhGswBtyM1eYh' },
            { 'PersonId': 'IN7MvYtUjkp1GMZu0F6CG9' }
        ]
    },
    'ValidFromDate' : `2017-09-14T`, --Ion timestamp literal with day precision
    'ValidToDate' : `2020-06-25T`
}
```

Backtick-Notation

PartiQL deckt alle Ion-Datentypen vollständig ab, sodass Sie jede Anweisung ohne Backticks schreiben können. Es gibt jedoch Fälle, in denen diese Ion-Literalsyntax Ihre Anweisungen klarer und prägnanter machen kann.

Um beispielsweise ein Dokument mit Ion-Zeitstempel- und Symbolwerten einzufügen, können Sie die folgende Anweisung nur mit der reinen PartiQL-Syntax schreiben.

```
INSERT INTO myTable VALUE
{
    'myTimestamp': to_timestamp('2019-09-04T'),
    'mySymbol': cast('foo' as symbol)
}
```

Dies ist ziemlich ausführlich, sodass Sie stattdessen Backticks verwenden können, um Ihre Anweisung zu vereinfachen.

```
INSERT INTO myTable VALUE
{
    'myTimestamp': `2019-09-04T`,
    'mySymbol': `foo`
}
```

Sie können auch die gesamte Struktur in Backticks einschließen, um einige weitere Tastatureingaben zu sparen.

```
INSERT INTO myTable VALUE
`{
    myTimestamp: 2019-09-04T,
    mySymbol: foo
}`
```

Important

Zeichenfolgen und Symbole sind unterschiedliche Klassen in PartiQL. Das bedeutet, dass sie, selbst wenn sie denselben Text haben, nicht gleichwertig sind. Die folgenden PartiQL-Ausdrücke werden beispielsweise zu verschiedenen Ion-Werten ausgewertet.

```
'foo'
```

```
`foo`
```

Pfadnavigation

Beim Schreiben von Data Manipulation Language(DML)-Anweisungen oder Abfrageanweisungen können Sie mithilfe von Pfadschritten auf Felder in verschachtelten Strukturen zugreifen. PartiQL unterstützt die Punktschreibweise für den Zugriff auf Feldnamen einer übergeordneten Struktur. Im folgenden Beispiel wird auf das `Model`-Feld eines übergeordneten Elements `Vehicle` zugegriffen.

```
Vehicle.Model
```

Zum Zugriff auf ein bestimmtes Element einer Liste können Sie den Operator Eckige Klammern verwenden, um eine Nicht null-basierte Ordnungszahl anzugeben. Im folgenden Beispiel wird auf das Element `SecondaryOwners` mit der Ordnungszahl 2 zugegriffen. Mit anderen Worten: Dies ist das dritte Element in der Liste.

```
SecondaryOwners[2]
```

Aliasing

QLDB unterstützt offene Inhalte und Schemas. Wenn Sie also auf bestimmte Felder in einer Anweisung zugreifen, können Sie am besten sicherstellen, dass Sie die erwarteten Ergebnisse erhalten, indem Sie Aliase verwenden. Zum Beispiel: Wenn Sie keinen expliziten Alias angeben, generiert das System einen impliziten Alias für Ihre `FROM`-Quellen.

```
SELECT VIN FROM Vehicle
--is rewritten to
```

```
SELECT Vehicle.VIN FROM Vehicle AS Vehicle
```

Aber die Ergebnisse sind aufgrund von Feldnamenkonflikten unvorhersehbar für Feldnamen. Wenn ein anderes Feld mit dem Namen `VIN` in einer verschachtelten Struktur innerhalb der Dokumente existiert, können die von dieser Abfrage zurückgegebenen `VIN`-Werte überraschend sein. Eine bewährte Methode ist die folgende Anweisung. Diese Abfrage deklariert `v` als Alias über der `Vehicle`-Tabelle. Das `AS`-Schlüsselwort ist optional.

```
SELECT v.VIN FROM Vehicle [ AS ] v
```

Aliasing ist besonders nützlich für Pfade in verschachtelten Sammlungen innerhalb eines Dokuments. Mit der folgenden Anweisung wird beispielsweise `o` als Alias über der Sammlung `VehicleRegistration.Owners` deklariert.

```
SELECT o.SecondaryOwners
FROM VehicleRegistration AS r, @r.Owners AS o
```

Das Zeichen `@` ist hier technisch optional. Es gibt jedoch explizit an, dass Sie die `Owners`-Struktur innerhalb von `VehicleRegistration` haben möchten, und nicht eine andere Sammlung mit dem Namen `Owners` (sofern vorhanden).

PartiQL-Spezifikation

Weitere Informationen zur PartiQL-Abfragesprache finden Sie unter PartiQL-Spezifikation.

PartiQL-Befehle in Amazon QLDB

PartiQL erweitert SQL-92, um Dokumente im Amazon Ion-Datenformat zu unterstützen. Amazon QLDB unterstützt die folgenden PartiQL-Befehle.

Informationen dazu, wie Sie den Zugriff steuern, um jeden PartiQL-Befehl für bestimmte Tabellen auszuführen, finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

> **Note**
>
> - QLDB unterstützt nicht alle PartiQL-Befehle.

- Alle PartiQL-Anweisungen in QLDB unterliegen Transaktionslimits, wie in definiertKontingente und Limits in Amazon QLDB.
- Diese Referenz enthält grundlegende Syntax- und Verwendungsbeispiele für PartiQL-Anweisungen, die Sie manuell auf der QLDB-Konsole oder der QLDB-Shell ausführen. Codebeispiele, die zeigen, wie ähnliche Anweisungen mit einer unterstützten Programmiersprache ausgeführt werden, finden Sie in den Tutorials unter Erste Schritte mit dem Treiber.

DDL-Anweisungen (Datendefinitionssprache)

Data Definition Language (DDL) ist die Gruppe von PartiQL-Anweisungen, die Sie zum Verwalten von Datenbankobjekten wie Tabellen und Indizes verwenden. Sie verwenden DDL, um diese Objekte zu erstellen und zu abzulegen.

- CREATE INDEX
- CREATE TABLE
- DROP INDEX
- DROP TABLE
- UNDROP TABLE

DML-Anweisungen (Datenmanipulationssprache)

Data Manipulation Language (DML) ist die Gruppe von PartiQL-Anweisungen, die Sie zum Verwalten von Daten in QLDB-Tabellen verwenden. Sie verwenden DML-Anweisungen, um Daten in einer Tabelle hinzuzufügen, zu ändern oder zu löschen.

Die folgenden Anweisungen für DML- und Abfragesprachen werden unterstützt:

- DELETE
- VON (INSERT, REMOVE oder SET)
- INSERT
- SELECT
- UPDATE

Befehl CREATE INDEX in Amazon QLDB

Verwenden Sie in Amazon QLDB den`CREATE INDEX` Befehl, um einen Index für ein Dokumentfeld in einer Tabelle zu erstellen.

Informationen zur Zugriffskontrolle zur Ausführung dieses PartiQL-Befehls für bestimmte Tabellen finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

> **Important**
>
> QLDB benötigt einen Index, um ein Dokument effizient nachschlagen zu können. Ohne Index muss QLDB beim Lesen von Dokumenten einen vollständigen Tabellenscan durchführen. Dies kann bei großen Tabellen zu Leistungsproblemen führen, einschließlich Parallelitätskonflikten und Transaktions-Timeouts.
> Um Tabellenscans zu vermeiden, müssen Sie Anweisungen mit einer`WHERE` Prädikatklausel unter Verwendung eines Gleichheitsoperators (=oder`IN`) für ein indiziertes Feld oder eine Dokument-ID ausführen. Weitere Informationen finden Sie unter Optimieren der Abfrageleistung.

Beachten Sie beim Erstellen von Indizes die folgenden Einschränkungen:

- Ein Index kann nur für ein einzelnes Feld der obersten Ebene erstellt werden. Zusammengesetzte, verschachtelte, eindeutige und funktionsbasierte Indizes werden nicht unterstützt.
- Sie können einen Index für alle Ion-Datentypen erstellen, einschließlich`list` und`struct`. Sie können die indizierte Suche jedoch nur nach Gleichheit des gesamten Ionenwerts durchführen, unabhängig vom Ionentyp. Wenn Sie beispielsweise einen`list` Typ als Index verwenden, können Sie keine indizierte Suche nach einem Element in der Liste durchführen.
- Die Abfrageleistung wird nur verbessert, wenn Sie ein Gleichheitsprädikat verwenden, z. B.`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`.

  QLDB berücksichtigt keine Ungleichungen in Abfrageprädikaten. Daher werden bereichsgefilterte Scans nicht implementiert.
- Bei indizierten Feldern muss die Groß- und Kleinschreibung beachtet werden. Sie dürfen höchstens 128 Zeichen lang sein.

- Die Indexerstellung in QLDB erfolgt asynchron. Die Zeit, die zum Erstellen eines Index für eine nicht leere Tabelle benötigt wird, hängt von der Tabellengröße ab. Weitere Informationen finden Sie unter Verwalten von Indexen.

Themen

Syntax

```
CREATE INDEX ON table_name (field)
```

Parameter

`table_name`

Der Name der Tabelle, in der Sie den Index erstellen möchten. Die Tabelle muss bereits vorhanden sein.

Beim Dateinamen muss die Groß- und Kleinschreibung beachtet werden.

`field`

Der Name des Dokumentfeldes, für das der Index erstellt werden soll. Das Feld muss ein Attribut der obersten Ebene sein.

Bei indizierten Feldern muss die Groß- und Kleinschreibung beachtet werden. Sie dürfen höchstens 128 Zeichen lang sein.

Sie können einen Index für alle Amazon Ion-Datentypen erstellen, einschließlich`list` und`struct`. Sie können die indizierte Suche jedoch nur nach Gleichheit des gesamten Ionenwerts durchführen, unabhängig vom Ionentyp. Wenn Sie beispielsweise einen`list` Typ als Index verwenden, können Sie keine indizierte Suche nach einem Element in der Liste durchführen.

Rückgabewert

`tableId`— Die eindeutige ID der Tabelle, für die Sie den Index erstellt haben.

Beispiele

```
CREATE INDEX ON VehicleRegistration (LicensePlateNumber)
```

```
CREATE INDEX ON Vehicle (VIN)
```

Programmgesteuertes Ausführen mit dem Treiber

Informationen zum programmgesteuerten Ausführen dieser Anweisung mithilfe des QLDB-Treibers finden Sie in den folgenden Tutorials unter Erste Schritte mit dem Treiber:

- Java:ErstErstErstErstErste-Erst |Kochbuchreferenz
- .NET:Erste-Schrit-T |Kochbuchreferenz
- Gehe:Schnellstart-Tutorial |Kochbuchreferenz
- Node.js:Erste-Schrit-T-T-TSchrit |Kochbuchreferenz — Referenz für
- Python:Erste Schritte |Kochbuchreferenz

CREATE TABLE-Befehl in Amazon QLDB

Verwenden Sie in Amazon QLDB den`CREATE TABLE` Befehl, um eine neue Tabelle zu erstellen.

Tabellen haben einfache Namen ohne Namespaces. QLDB unterstützt offene Inhalte und erzwingt kein Schema, sodass Sie beim Erstellen von Tabellen keine Attribute oder Datentypen definieren.

> **Note**
>
> Informationen zur Zugriffskontrolle zur Ausführung dieses PartiQL-Befehls in einem Ledger finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

Themen

- Syntax

Syntax

```
CREATE TABLE table_name [ WITH (aws_tags = `{'key': 'value'}`) ]
```

Parameter

table_name

Der eindeutige Name der Tabelle, die erstellt werden soll. Eine aktive Tabelle mit dem gleichen Namen darf noch nicht existieren. Im Folgenden finden Sie die Benennungseinschränkungen:

- Darf nur 1—128 alphanumerische Zeichen oder Unterstriche enthalten.
- Muss einen Groß- oder Kleinbuchstaben für das erste Zeichen enthalten.
- Kann eine beliebige Kombination aus alphanumerischen Zeichen und Unterstrichen für die verbleibenden Zeichen enthalten.
- Beachten Sie die Groß- und Kleinschreibung.
- Darf kein reserviertes QLDB PartiQL-Wort sein.

'***Schlüssel***': '***Wert***'

(Optional) Die Tags, die bei der Erstellung an die Tabellenressource angefügt werden sollen. Jedes Tag ist als Schlüssel-Wert-Paar definiert, wobei der Schlüssel und der Wert jeweils durch einfache Anführungszeichen gekennzeichnet sind. Jedes Schlüssel-Wert-Paar ist innerhalb einer Amazon Ion-Struktur definiert, die durch Backticks gekennzeichnet ist.

Das Markieren von Tabellen bei der Erstellung wird derzeit nur für Ledger im*STANDARD* Berechtigungsmodus unterstützt.

Rückgabewert

`tableId`— Die eindeutige ID der Tabelle, die Sie erstellt haben.

Markieren von Tabellen bei der Erstellung

> **Note**
>
> Das Markieren von Tabellen bei der Erstellung wird derzeit nur für Ledger im`STANDARD` Berechtigungsmodus unterstützt.

Optional können Sie Ihre Tabellenressourcen taggen, indem Sie Tags in einer`CREATE TABLE` Anweisung angeben. Weitere Informationen zu Tags erhalten Sie unter Markieren von Amazon-QLDB Ressourcen. Im folgenden Beispiel wird eine Tabelle erstellt, die`Vehicle` mit dem Tag benannt ist`environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Für das Markieren von Tabellen bei der Erstellung ist der Zugriff auf die`qldb:TagResource` Aktionen`qldb:PartiQLCreateTable` und erforderlich. Weitere Informationen zu Berechtigungen für QLDB -Ressourcen finden Sie unterFunktionsweise von Amazon QLDB mit IAM.

Indem Sie Ressourcen zum Erstellungszeitpunkt markieren, müssen Sie anschließend keine benutzerdefinierten Markierungs-Skripts ausführen. Nachdem eine Tabelle mit Tags versehen wurde, können Sie den Zugriff auf die Tabelle anhand dieser Tags steuern. Sie können beispielsweise vollen Zugriff nur auf Tabellen gewähren, die ein bestimmtes Tag haben. Ein Beispiel für eine JSON-Richtlinie finden Sie unterVollzugriff auf alle Aktionen basierend auf Tabellen-Tags.

Beispiele

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'development'}`)
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'key1': 'value1', 'key2': 'value2'}`)
```

Programmgesteuertes Ausführen mit dem Treiber

Informationen zum programmgesteuerten Ausführen dieser Anweisung mithilfe des QLDB-Treibers finden Sie in den folgenden Tutorials unter Erste Schritte mit dem Treiber:

- Java:ErstErstErstErstErste-Erst |Kochbuchreferenz
- .NET:Erste-Schrit-T |Kochbuchreferenz
- Gehe:Schnellstart-Tutorial |Kochbuchreferenz
- Node.js:Erste-Schrit-T-T-TSchrit |Kochbuchreferenz — Referenz für
- Python:Erste Schritte |Kochbuchreferenz

Befehl DELETE in Amazon QLDB

Verwenden Sie in Amazon QLDB den`DELETE` Befehl, um ein aktives Dokument in einer Tabelle als gelöscht zu markieren, indem Sie eine neue, aber endgültige Version des Dokuments erstellen. Diese endgültige Überarbeitung bedeutet, dass das Dokument gelöscht wurde. Dieser Vorgang beendet den Lebenszyklus eines Dokuments, was bedeutet, dass keine weiteren Dokumentrevisionen mit derselben Dokument-ID erstellt werden können.

Dieser Vorgang ist irreversibel. Sie können den Revisionsverlauf eines gelöschten Dokuments weiterhin abfragen, indem Sie den verwendenVerlaufsfunktion.

> **Note**
>
> Informationen zur Zugriffskontrolle zur Ausführung dieses PartiQL-Befehls für bestimmte Tabellen finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

Themen

- Syntax
- Parameter
- Rückgabewert
- Beispiele
- Programmgesteuertes Ausführen mit dem Treiber

Syntax

```
DELETE FROM table_name [ AS table_alias ] [ BY id_alias ]
```

```
[ WHERE condition ]
```

Parameter

table_name

Der Name der Benutzertabelle mit den zu löschenden Daten. DML-Anweisungen werden nur in der Standard-Benutzeransicht unterstützt. Jede Anweisung kann nur auf einer einzigen Tabelle ausgeführt werden.

AS ***table_alias***

(Optional) Ein benutzerdefinierter Alias, der sich über eine Tabelle erstreckt, aus der gelöscht werden soll. Das `AS`-Schlüsselwort ist optional.

BY ***id_alias***

(Optional) Eine benutzerdefinierter Alias, der an das Metadatenfeld `id` für jedes Dokument im Ergebnissatz gebunden ist. Der Alias muss in der `FROM`-Klausel mit dem Schlüsselwort `BY` deklariert werden. Dies ist nützlich, wenn Sie nach der Dokument-ID filtern möchten, während Sie die Standard-Benutzeransicht abfragen. Weitere Informationen finden Sie unter Verwenden der BY-Klausel zur Abfrage der Dokument-ID.

WHERE ***condition***

Die Auswahlkriterien für die Dokumente, die gelöscht werden sollen.

> **Note**
>
> Wenn Sie die `WHERE`-Klausel weglassen, werden alle Dokumente in der Tabelle gelöscht.

Rückgabewert

`documentId`— Die eindeutige ID jedes Dokuments, das Sie gelöscht haben.

Beispiele

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

Programmgesteuertes Ausführen mit dem Treiber

Informationen zum programmgesteuerten Ausführen dieser Anweisung mithilfe des QLDB-Treibers finden Sie in den folgenden Tutorials unter Erste Schritte mit dem Treiber:

- Java:ErstErstErstErstErste-Erst |Kochbuchreferenz
- .NET:Erste-Schrit-T |Kochbuchreferenz
- Gehe:Schnellstart-Tutorial |Kochbuchreferenz
- Node.js:Erste-Schrit-T-T-TSchrit |Kochbuchreferenz — Referenz für
- Python:Erste Schritte |Kochbuchreferenz

Befehl DROP INDEX in Amazon QLDB

Verwenden Sie in Amazon QLDB den`DROP INDEX` Befehl, um einen Index für eine Tabelle zu löschen.

> **Note**
>
> Informationen zur Zugriffskontrolle zur Ausführung dieses PartiQL-Befehls für bestimmte Tabellen finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

Themen

- Syntax
- Parameter
- Rückgabewert
- Beispiele

Syntax

```
DROP INDEX "indexId" ON table_name WITH (purge = true)
```

> **Note**
>
> Die Klausel`WITH (purge = true)` ist für alle`DROP INDEX` Anweisungen erforderlich und`true` ist derzeit der einzige unterstützte Wert.
> Bei diesem`purge` Schlüsselwort muss die Groß- und Kleinschreibung beachtet werden.

Parameter

„*`Index-ID`*“

Die eindeutige ID des zu löschenden Indexes, gekennzeichnet durch doppelte Anführungszeichen.

KEIN *`Tabellenname`*

Der Name der Tabelle, deren Index Sie löschen möchten.

Rückgabewert

`tableId`— Die eindeutige ID der Tabelle, deren Index Sie gelöscht haben.

Beispiele

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

DROP TABLE-Befehl in Amazon QLDB

Verwenden Sie in Amazon QLDB den`DROP TABLE` Befehl, um eine vorhandene Tabelle zu deaktivieren. Sie können die Anweisung UNDROP TABLE verwenden, um sie erneut zu aktivieren. Das Deaktivieren oder Reaktivieren einer Tabelle hat keine Auswirkungen auf ihre Dokumente oder Indizes.

> **Note**
>
> Informationen zur Zugriffskontrolle zur Ausführung dieses PartiQL-Befehls für bestimmte Tabellen finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

Themen

Syntax

```
DROP TABLE table_name
```

Parameter

`table_name`

Der Name der Tabelle, die deaktiviert werden soll. Diese Tabelle muss bereits vorhanden sein und den Status `ACTIVE` haben.

Rückgabewert

`tableId`— Die eindeutige ID der Tabelle, die Sie deaktiviert haben.

Beispiele

```
DROP TABLE VehicleRegistration
```

Befehl FROM (INSERT, REMOVE oder SET) in Amazon QLDB

In Amazon QLDB`FROM` ist eine Anweisung, die mit beginnt, eine PartiQL-Erweiterung, mit der Sie bestimmte Elemente in ein Dokument einfügen und entfernen können. Sie können diese Anweisung auch zum Aktualisieren von vorhandenen Elementen in einem Dokument verwenden, ähnlich demUPDATE Befehl.

> **Note**
>
> Informationen zur Zugriffskontrolle zur Ausführung dieses PartiQL-Befehls für bestimmte Tabellen finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

Themen

Syntax

VON - EINFÜGEN

Fügen Sie ein neues Element in ein vorhandenes Dokument ein. Zum Einfügen eines neuen Dokuments in eine Tabelle müssen Sie INSERT verwenden.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
INSERT INTO element VALUE data [ AT key_name ]
```

VON-ENTFERNEN

Entfernen Sie ein vorhandenes Element in einem Dokument oder entfernen Sie ein gesamtes Dokument der oberen Ebene. Letzteres ist semantisch identisch mit der traditionellen DELETE-Syntax.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
REMOVE element
```

AB SET

Aktualisieren Sie ein oder mehrere Elemente in einem Dokument. Wenn ein Element nicht existiert, wird es eingefügt. Dies ist semantisch identisch mit der traditionellen UPDATE-Syntax.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
SET element = data [, element = data, ... ]
```

Parameter

table_name

Der Name der Benutzertabelle mit den zu ändernden Daten. DML-Anweisungen werden nur in der Standard-Benutzeransicht unterstützt. Jede Anweisung kann nur auf einer einzigen Tabelle ausgeführt werden.

In dieser Klausel können Sie auch eine oder mehrere Sammlungen einschließen, die in der angegebenen Tabelle verschachtelt sind. Weitere Details finden Sie unter Verschachtelte Sammlungen.

AS ***table_alias***

(Optional) Ein benutzerdefinierter Alias, der sich über eine zu ändernde Tabelle erstreckt. Alle Tabellenaliasse, die in der Klausel `SET`, `REMOVE`, `INSERT INTO` oder `WHERE` verwendet werden, müssen in der `FROM`-Klausel deklariert werden. Das `AS`-Schlüsselwort ist optional.

BY ***id_alias***

(Optional) Eine benutzerdefinierter Alias, der an das Metadatenfeld `id` für jedes Dokument im Ergebnissatz gebunden ist. Der Alias muss in der `FROM`-Klausel mit dem Schlüsselwort `BY` deklariert werden. Dies ist nützlich, wenn Sie nach der Dokument-ID filtern möchten, während Sie die Standard-Benutzeransicht abfragen. Weitere Informationen finden Sie unter Verwenden der BY-Klausel zur Abfrage der Dokument-ID.

WHERE ***condition***

Die Auswahlkriterien für die Dokumente, die geändert werden sollen.

> **Note**
>
> Wenn Sie die `WHERE`-Klausel weglassen, werden alle Dokumente in der Tabelle geändert.

element

Ein Dokumentelement, das erstellt oder geändert werden soll.

data

Ein neuer Wert für das Element.

AT ***key_name***

Ein Schlüsselname, der innerhalb des zu ändernden Dokuments hinzugefügt werden soll. Sie müssen den entsprechenden `VALUE` zusammen mit den Schlüsselnamen angeben. Dies ist für das Einfügen eines neuen Werts `AT` einer bestimmten Position innerhalb eines Dokuments erforderlich.

Verschachtelte Sammlungen

Sie können eine DML-Anweisung zwar nur für eine einzelne Tabelle ausführen, Sie können jedoch verschachtelte Sammlungen innerhalb von Dokumenten in dieser Tabelle als zusätzliche Quellen angeben. Jeder Alias, den Sie für eine verschachtelte Auflistung deklarieren, kann in der `WHERE`-Klausel und der `SET`-, `INSERT INTO`- oder `REMOVE`-Klausel verwendet werden.

Beispielsweise umfassen die `FROM`-Quellen der folgenden Anweisung sowohl die `VehicleRegistration`-Tabelle als auch die verschachtelte `Owners.SecondaryOwners`-Struktur.

```
FROM VehicleRegistration r, @r.Owners.SecondaryOwners o
WHERE r.VIN = '1N4AL11D75C109151' AND o.PersonId = 'abc123'
SET o.PersonId = 'def456'
```

In diesem Beispiel wird das spezifische Element der Liste `SecondaryOwners` aktualisiert, das als `PersonId 'abc123'` innerhalb des Dokuments `VehicleRegistration` mit der `VIN '1N4AL11D75C109151'` enthält. Mit diesem Ausdruck können Sie ein Element einer Liste anhand seines Werts und nicht anhand seines Indexes angeben.

Rückgabewert

`documentId`— Die eindeutige ID jedes Dokuments, das Sie aktualisiert oder gelöscht haben.

Beispiele

Ändern Sie ein Element in einem Dokument. Wenn das -Element nicht vorhanden ist, wird es eingefügt.

```
FROM Vehicle AS v
WHERE v.VIN = '1N4AL11D75C109151' AND v.Color = 'Silver'
SET v.Color = 'Shiny Gray'
```

Ändern Sie ein Element oder fügen Sie ein Element ein und filtern Sie nach dem vom System zugewiesenen Dokument-Metadatenfeld `id`.

```
FROM Vehicle AS v BY v_id
WHERE v_id = 'documentId'
SET v.Color = 'Shiny Gray'
```

Ändern Sie das `PersonId`-Feld des ersten Elements in der `Owners.SecondaryOwners`-Liste innerhalb eines Dokuments.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
```

Entfernen Sie ein vorhandenes Element in einem Dokument.

```
FROM Person AS p
WHERE p.GovId = '111-22-3333'
REMOVE p.Address
```

Entfernen Sie ein ganzes Dokument aus einer Tabelle.

```
FROM Person AS p
WHERE p.GovId = '111-22-3333'
REMOVE p
```

Entfernen Sie das erste Element in der `Owners.SecondaryOwners`-Liste innerhalb eines Dokuments in der `VehicleRegistration`-Tabelle.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
REMOVE r.Owners.SecondaryOwners[0]
```

`{'Mileage':26500}`Als Name-Wert-Paar der obersten Ebene innerhalb eines Dokuments in die`Vehicle` Tabelle einfügen.

```
FROM Vehicle AS v
WHERE v.VIN = '1N4AL11D75C109151'
```

```
INSERT INTO v VALUE 26500 AT 'Mileage'
```

Fügen Sie `{'PersonId':'abc123'}` als Name-Wert-Paar in das `Owners.SecondaryOwners`-Feld eines Dokuments in der-`VehicleRegistration` Tabelle an. Beachten Sie, dass `Owners.SecondaryOwners` bereits vorhanden und ein Listen-Datentyp sein muss, damit diese Anweisung gültig ist. Andernfalls ist das Schlüsselwort `AT` in der `INSERT INTO`-Klausel erforderlich.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
```

Fügen Sie `{'PersonId':'abc123'}` als erstes Element in der vorhandenen `Owners.SecondaryOwners`-Liste innerhalb eines Dokuments ein.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
```

Hängen Sie mehrere Name-Wert-Paare an die bestehende `Owners.SecondaryOwners`-Liste innerhalb eines Dokuments an.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
```

Programmgesteuertes Ausführen mit dem Treiber

Informationen zum programmgesteuerten Ausführen dieser Anweisung mithilfe des QLDB-Treibers finden Sie in den folgenden Tutorials unter Erste Schritte mit dem Treiber:

- Java:ErstErstErstErstErste-Erst |Kochbuchreferenz
- .NET:Erste-Schrit-T |Kochbuchreferenz
- Gehe:Schnellstart-Tutorial |Kochbuchreferenz
- Node.js:Erste-Schrit-T-T-TSchrit |Kochbuchreferenz — Referenz für
- Python:Erste Schritte |Kochbuchreferenz

INSERT-Befehl in Amazon QLDB

Verwenden Sie in Amazon QLDB den`INSERT` Befehl, um einer Tabelle ein oder mehrere Amazon Ion-Dokumente hinzuzufügen.

> **Note**
>
> Informationen zur Zugriffskontrolle zur Ausführung dieses PartiQL-Befehls für bestimmte Tabellen finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

Themen

Syntax

Fügen Sie ein einzelnes Dokument ein.

```
INSERT INTO table_name VALUE document
```

Fügen Sie mehrere Dokumente ein.

```
INSERT INTO table_name << document, document, ... >>
```

Parameter

table_name

Der Name der Benutzertabelle, in der Sie die Daten einfügen möchten. Die Tabelle muss bereits vorhanden sein. DML-Anweisungen werden nur in der Standard-Benutzeransicht unterstützt.

document

Ein gültiges QLDB-Dokument. Sie müssen mindestens ein Dokument angeben. Mehrere Dokumente müssen durch Kommas getrennt werden.

Das Dokument muss mit geschweiften Klammern (`{...}`) gekennzeichnet sein.

Jeder Feldname im Dokument ist ein Ionensymbol, bei dem zwischen Groß- und Kleinschreibung unterschieden wird, das in PartiQL durch einfache Anführungszeichen (`'...'`) gekennzeichnet werden kann.

Zeichenkettenwerte werden in PartiQL auch durch einfache Anführungszeichen (`'...'`) gekennzeichnet.

Alle Ion-Literalwerte können mit umgekehrten Anführungszeichen angegeben werden (`` `...` ``).

> **Note**
>
> Doppelte eckige Klammern (`<<...>>`) bezeichnen eine ungeordnete Sammlung (in PartiQL als Tasche bezeichnet) und sind nur erforderlich, wenn Sie mehrere Dokumente einfügen möchten.

Rückgabewert

`documentId`— Die eindeutige ID jedes Dokuments, das Sie eingefügt haben.

Beispiele

Fügen Sie ein einzelnes Dokument ein.

```
INSERT INTO VehicleRegistration VALUE
{
    'VIN' : 'KM8SRDHF6EU074761', --string
    'RegNum' : 1722, --integer
    'State' : 'WA',
    'City' : 'Kent',
    'PendingPenaltyTicketAmount' : 130.75, --decimal
    'Owners' : { --nested struct
        'PrimaryOwner' : { 'PersonId': '294jJ3YUoH1IEEm8GSabOs' },
        'SecondaryOwners' : [ --list of structs
            { 'PersonId' : '1nmeDdLo3AhGswBtyM1eYh' },
```

```
            { 'PersonId': 'IN7MvYtUjkp1GMZu0F6CG9' }
        ]
    },
    'ValidFromDate' : `2017-09-14T`, --Ion timestamp literal with day precision
    'ValidToDate' : `2020-06-25T`
}
```

Diese Anweisung gibt die eindeutige ID des von Ihnen eingefügten Dokuments wie folgt zurück.

```
{
    documentId: "2kKuOPNB07D2iTPBrUTWGl"
}
```

Fügen Sie mehrere Dokumente ein.

```
INSERT INTO Person <<
{
    'FirstName' : 'Raul',
    'LastName' : 'Lewis',
    'DOB' : `1963-08-19T`,
    'GovId' : 'LEWISR261LL',
    'GovIdType' : 'Driver License',
    'Address' : '1719 University Street, Seattle, WA, 98109'
},
{
    'FirstName' : 'Brent',
    'LastName' : 'Logan',
    'DOB' : `1967-07-03T`,
    'GovId' : 'LOGANB486CG',
    'GovIdType' : 'Driver License',
    'Address' : '43 Stockert Hollow Road, Everett, WA, 98203'
},
{
    'FirstName' : 'Alexis',
    'LastName' : 'Pena',
    'DOB' : `1974-02-10T`,
    'GovId' : '744 849 301',
    'GovIdType' : 'SSN',
    'Address' : '4058 Melrose Street, Spokane Valley, WA, 99206'
}
>>
```

Diese Anweisung gibt die eindeutige ID jedes von Ihnen eingefügten Dokuments wie folgt zurück.

```
{
    documentId: "6WXzLscsJ3bDWW97Dy8nyp"
},
{
    documentId: "35e0ToZyTGJ7LGvcwrkX65"
},
{
    documentId: "BVHPcH612o7JROQ4yP8jiH"
}
```

Programmgesteuertes Ausführen mit dem Treiber

Informationen zum programmgesteuerten Ausführen dieser Anweisung mithilfe des QLDB-Treibers finden Sie in den folgenden Tutorials unter Erste Schritte mit dem Treiber:

- Java:ErstErstErstErstErste-Erst |Kochbuchreferenz
- .NET:Erste-Schrit-T |Kochbuchreferenz
- Gehe:Schnellstart-Tutorial |Kochbuchreferenz
- Node.js:Erste-Schrit-T-T-TSchrit |Kochbuchreferenz — Referenz für
- Python:Erste Schritte |Kochbuchreferenz

SELECT-Befehl in Amazon QLDB

Verwenden Sie in Amazon QLDB den`SELECT` Befehl, um Daten aus einer oder mehreren Tabellen abzurufen. Jede`SELECT` Abfrage in QLDB wird in einer Transaktion verarbeitet und unterliegt einem Transaktions-Timeout-Limit.

Die Reihenfolge der Ergebnisse ist nicht spezifisch und kann für jede`SELECT` Abfrage variieren. Sie sollten sich bei keiner Abfrage in QLDB auf die Reihenfolge der Ergebnisse verlassen.

Informationen zur Zugriffskontrolle zur Ausführung dieses PartiQL-Befehls für bestimmte Tabellen finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

> **Warning**
>
> Wenn Sie eine Abfrage in QLDB ohne indizierte Suche ausführen, wird ein vollständiger Tabellenscan aufgerufen. PartiQL unterstützt solche Abfragen, da es SQL-kompatibel ist. Führen Sie jedoch keine Tabellenscans für produktive Anwendungsfälle in QLDB aus.

Tabellenscans können bei großen Tabellen zu Leistungsproblemen führen, einschließlich Parallelitätskonflikten und Transaktions-Timeouts.
Um Tabellenscans zu vermeiden, müssen Sie Anweisungen mit einer`WHERE` Prädikatklausel ausführen, die einen Gleichheitsoperator für ein indiziertes Feld oder eine Dokument-ID verwenden, z. B.`WHERE indexedField = 123` oder`WHERE indexedField IN (456, 789)`. Weitere Informationen finden Sie unter Optimieren der Abfrageleistung.

Themen

- Syntax
- Parameter
- Joins
- Verschachtelte Abfragen — Einschränkungen
- Beispiele
- Programmgesteuertes Ausführen mit dem Treiber

Syntax

```
SELECT [ VALUE ] expression [ AS field_alias ] [, expression, ... ]
FROM source [ AS source_alias ] [ AT idx_alias ] [ BY id_alias ] [, source, ... ]
[ WHERE condition ]
```

Parameter

VALUE

Ein Qualifizierer für Ihren Ausdruck, durch den die Abfrage den unformatierten Datentypwert zurückgibt, anstatt den Wert, der von einer Tupel-Struktur umgeben ist.

Ausdruck

Eine Projektion aus dem *-Platzhalter oder eine Projektionsliste aus einem oder mehreren Dokumentfeldern aus dem Ergebnissatz. Ein Ausdruck kann aus Aufrufen an PartiQL-Funktionen oder Feldern bestehen, die von PartiQL-Operatoren geändert werden.

AS ***field_alias***

(Optional) Ein temporärer, benutzerdefinierter Alias für das Feld, das in dem endgültigen Ergebnissatz verwendet wird. Das AS-Schlüsselwort ist optional.

Wenn Sie kein Alias für einen Ausdruck angeben, bei dem es sich nicht um einen einfachen Feldnamen handelt, wendet der Ergebnissatz einen Standardnamen auf dieses Feld an.

AUS der ***Quelle***

Eine Quelle, die abgefragt werden soll. Die einzigen derzeit unterstützten Quellen sind Tabellennamen, interne Joins zwischen Tabellen, verschachtelte `SELECT`-Abfragen (vorbehaltlich Verschachtelte Abfragen — Einschränkungen) und Verlaufsfunktionsaufrufe für eine Tabelle.

Sie müssen mindestens eine Quelle angeben. Mehrere Quellen müssen durch Kommata getrennt werden.

AS ***source_alias***

Ein optionaler, benutzerdefinierter Alias-Name, der über eine Quelle reicht, aus der abgefragt werden soll. Alle Quell-Aliasse, die in den Klauseln `SELECT` ODER `WHERE` verwendet werden, müssen in der `FROM`-Klausel deklariert werden. Das `AS`-Schlüsselwort ist optional.

AT ***idx_alias***

(Optional) Ein benutzerdefinierter Alias, der an die Indexnummer (Ordinalnummer) jedes Elements innerhalb einer Liste aus der Quelle bindet. Der Alias muss in der `FROM`-Klausel mit dem Schlüsselwort `AT` deklariert werden.

BY ***id_alias***

(Optional) Eine benutzerdefinierter Alias, der an das Metadatenfeld `id` für jedes Dokument im Ergebnissatz gebunden ist. Der Alias muss in der `FROM`-Klausel mit dem Schlüsselwort `BY` deklariert werden. Dies ist nützlich, wenn Sie nach der Dokument-ID projizieren oder filtern möchten, während Sie die Standard-Benutzeransicht abfragen. Weitere Informationen finden Sie unter Verwenden der BY-Klausel zur Abfrage der Dokument-ID.

WHERE ***condition***

Die Auswahlkriterien und Join-Kriterien (falls zutreffend) für die Abfrage.

> **Note**
>
> Wenn Sie die `WHERE`-Klausel weglassen, werden alle Dokumente in der Tabelle abgerufen.

Joins

Nur innere Joins werden derzeit unterstützt. Sie können Abfragen mithilfe der expliziten `INNER JOIN`-Klausel innere Join-Abfragen wie folgt schreiben. In dieser Syntax muss `JOIN` mit `ON` gekoppelt werden und das `INNER`-Schlüsselwort ist optional.

```
SELECT expression
FROM table1 AS t1 [ INNER ] JOIN table2 AS t2
ON t1.element = t2.element
```

Sie können auch wie folgt innere Joins mithilfe der impliziten Syntax schreiben.

```
SELECT expression
FROM table1 AS t1, table2 AS t2
WHERE t1.element = t2.element
```

Verschachtelte Abfragen — Einschränkungen

Sie können verschachtelte Abfragen innerhalb von `SELECT`-Ausdrücken und innerhalb von `FROM`-Quellen schreiben. Die wichtigste Beschränkung besteht darin, dass nur die äußerste Abfrage auf die globale Datenbankumgebung zugreifen kann. Beispiel: Angenommen, Sie verfügen über einen Ledger mit den Tabellen `VehicleRegistration` und `Person`. Die folgende verschachtelte Abfrage ist nicht gültig, weil die innere `SELECT` versucht, auf `Person` zuzugreifen.

```
SELECT r.VIN,
    (SELECT p.PersonId FROM Person AS p WHERE p.PersonId =
 r.Owners.PrimaryOwner.PersonId) AS PrimaryOwner
FROM VehicleRegistration AS r
```

Die folgende verschachtelte Abfrage ist allerdings gültig.

```
SELECT r.VIN, (SELECT o.PrimaryOwner.PersonId FROM @r.Owners AS o) AS PrimaryOwner
FROM VehicleRegistration AS r
```

Beispiele

Die folgende Abfrage zeigt einen einfachen Platzhalter für`SELECT` alle mit einer`WHERE` Standardprädikatklausel, die den`IN` Operator verwendet.

```
SELECT * FROM Vehicle
WHERE VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Das folgende Beispiel zeigt SELECT-Projektionen mit einem Zeichenfolgenfilter.

```
SELECT FirstName, LastName, Address
FROM Person
WHERE Address LIKE '%Seattle%'
AND GovId = 'LEWISR261LL'
```

Das folgende Beispiel zeigt eine korrelierte Unterabfrage, die verschachtelte Daten vereinfacht. Beachten Sie, dass das Zeichen @ hier technisch optional ist. Es gibt jedoch ausdrücklich an, dass Sie die darin verschachtelte`Owners` Struktur verwenden möchten`VehicleRegistration`, nicht einen anderen Namen für die Sammlung`Owners` (falls vorhanden). Weitere Informationen finden SieNested data im Kapitel Arbeiten mit Daten und Historie.

```
SELECT
    r.VIN,
    o.SecondaryOwners
FROM
    VehicleRegistration AS r, @r.Owners AS o
WHERE
    r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Das folgende Beispiel zeigt eine Unterabfrage in der SELECT-Liste, die verschachtelte Daten vereinfacht, sowie einen impliziten inneren Join.

```
SELECT
    v.Make,
    v.Model,
    (SELECT VALUE o.PrimaryOwner.PersonId FROM @r.Owners AS o) AS PrimaryOwner
FROM
    VehicleRegistration AS r, Vehicle AS v
WHERE
    r.VIN = v.VIN AND r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Das folgende Beispiel zeigt einen expliziten inneren Join.

```
SELECT
    v.Make,
```

```
    v.Model,
    r.Owners
FROM
    VehicleRegistration AS r JOIN Vehicle AS v
ON
    r.VIN = v.VIN
WHERE
    r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Das folgende Beispiel zeigt eine Projektion des Dokument-`id`-Metadatenfelds unter Verwendung der BY-Klausel.

```
SELECT
    r_id,
    r.VIN
FROM
    VehicleRegistration AS r BY r_id
WHERE
    r_id = 'documentId'
```

Im Folgenden wird dieBY Klausel verwendet, um die`Person` Tabellen`DriversLicense` und in ihren`PersonId` jeweiligen`id` Dokumentfeldern zu verknüpfen.

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
WHERE pid = 'documentId'
```

Im Folgenden werden die verwendetEngagierter Standpunkt, um die`Person` Tabellen`DriversLicense` und in ihren`PersonId` jeweiligen`id` Dokumentfeldern miteinander zu verbinden.

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS cp
ON d.PersonId = cp.metadata.id
WHERE cp.metadata.id = 'documentId'
```

Im Folgenden wird die Ordnungszahl`PersonId` und die Indexnummer (Ordinalzahl) jeder Person in der`Owners.SecondaryOwners` Liste für ein Dokument in der Tabelle zurückgegeben`VehicleRegistration`.

```
SELECT s.PersonId, owner_idx
```

```
FROM VehicleRegistration AS r, @r.Owners.SecondaryOwners AS s AT owner_idx
WHERE r.VIN = 'KM8SRDHF6EU074761'
```

Programmgesteuertes Ausführen mit dem Treiber

Informationen zum programmgesteuerten Ausführen dieser Anweisung mithilfe des QLDB-Treibers finden Sie in den folgenden Tutorials unter Erste Schritte mit dem Treiber:

- Java:ErstErstErstErstErste-Erst |Kochbuchreferenz
- .NET:Erste-Schrit-T |Kochbuchreferenz
- Gehe:Schnellstart-Tutorial |Kochbuchreferenz
- Node.js:Erste-Schrit-T-T-TSchrit |Kochbuchreferenz — Referenz für
- Python:Erste Schritte |Kochbuchreferenz

Befehl AKTUALISIEREN in Amazon QLDB

Verwenden Sie in Amazon QLDB den`UPDATE` Befehl, um den Wert eines oder mehrerer Elemente innerhalb eines Dokuments zu ändern. Wenn ein Element nicht existiert, wird es eingefügt.

Sie können diesen Befehl auch verwenden, um bestimmte Elemente in einem Dokument explizit einzufügen und zu entfernen, ähnlich wie beiVON (INSERT, REMOVE oder SET) Anweisungen.

> **Note**
>
> Informationen zur Zugriffskontrolle zur Ausführung dieses PartiQL-Befehls für bestimmte Tabellen finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

Themen

Syntax

AKTUALISIERUNGSSATZ

Aktualisieren Sie ein oder mehrere Elemente in einem Dokument. Wenn ein Element nicht existiert, wird es eingefügt. Dies entspricht semantisch der FROM-SET-Anweisung.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
SET element = data [, element = data, ... ]
[ WHERE condition ]
```

AKTUALISIEREN-EINFÜGEN

Fügen Sie ein neues Element in ein vorhandenes Dokument ein. Zum Einfügen eines neuen Dokuments in eine Tabelle müssen Sie INSERT verwenden.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
INSERT INTO element VALUE data [ AT key_name ]
[ WHERE condition ]
```

AKTUALISIEREN-ENTFERNEN

Entfernen Sie ein vorhandenes Element in einem Dokument oder entfernen Sie ein gesamtes Dokument der oberen Ebene. Letzteres ist semantisch identisch mit der traditionellen DELETE-Syntax.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
REMOVE element
[ WHERE condition ]
```

Parameter

table_name

Der Name der Benutzertabelle mit den zu ändernden Daten. DML-Anweisungen werden nur in der Standard-Benutzeransicht unterstützt. Jede Anweisung kann nur auf einer einzigen Tabelle ausgeführt werden.

AS ***table_alias***

(Optional) Ein benutzerdefinierter Alias, der sich über eine zu aktualisierende Tabelle erstreckt. Das AS-Schlüsselwort ist optional.

BY *id_alias*

(Optional) Eine benutzerdefinierter Alias, der an das Metadatenfeld `id` für jedes Dokument im Ergebnissatz gebunden ist. Der Alias muss in der `UPDATE`-Klausel mit dem Schlüsselwort `BY` deklariert werden. Dies ist nützlich, wenn Sie nach der Dokument-ID filtern möchten, während Sie die Standard-Benutzeransicht abfragen. Weitere Informationen finden Sie unter Verwenden der BY-Klausel zur Abfrage der Dokument-ID.

element

Ein Dokumentelement, das erstellt oder geändert werden soll.

data

Ein neuer Wert für das Element.

AT *key_name*

Ein Schlüsselname, der innerhalb des zu ändernden Dokuments hinzugefügt werden soll. Sie müssen den entsprechenden `VALUE` zusammen mit den Schlüsselnamen angeben. Dies ist für das Einfügen eines neuen Werts `AT` einer bestimmten Position innerhalb eines Dokuments erforderlich.

WHERE *condition*

Die Auswahlkriterien für die Dokumente, die geändert werden sollen.

> **Note**
>
> Wenn Sie die `WHERE`-Klausel weglassen, werden alle Dokumente in der Tabelle geändert.

Rückgabewert

`documentId`— Die eindeutige ID jedes Dokuments, das Sie aktualisiert haben.

Beispiele

Aktualisieren Sie ein Feld in einem Dokument. Wenn das Feld nicht vorhanden ist, wird es eingefügt.

```
UPDATE Person AS p
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE p.GovId = '111-22-3333'
```

Filtern Sie das vom System zugewiesene Dokument-`id`-Metadatenattribut.

```
UPDATE Person AS p BY pid
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE pid = 'documentId'
```

Überschreiben Sie ein gesamtes Dokument.

```
UPDATE Person AS p
SET p = {
    'FirstName' : 'Rosemarie',
    'LastName' : 'Holloway',
    'DOB' : `1977-06-18T`,
    'GovId' : '111-22-3333',
    'GovIdType' : 'Driver License',
    'Address' : '4637 Melrose Street, Ellensburg, WA, 98926'
}
WHERE p.GovId = '111-22-3333'
```

Ändern Sie das `PersonId`-Feld des ersten Elements in der `Owners.SecondaryOwners`-Liste innerhalb eines Dokuments.

```
UPDATE VehicleRegistration AS r
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
WHERE r.VIN = '1N4AL11D75C109151'
```

`{'Mileage':26500}`Als Name-Wert-Paar der obersten Ebene innerhalb eines Dokuments in die`Vehicle` Tabelle einfügen.

```
UPDATE Vehicle AS v
INSERT INTO v VALUE 26500 AT 'Mileage'
WHERE v.VIN = '1N4AL11D75C109151'
```

Fügen Sie `{'PersonId':'abc123'}` als Name-Wert-Paar in das `Owners.SecondaryOwners`-Feld eines Dokuments in der`VehicleRegistration` Tabelle an. Beachten Sie, dass `Owners.SecondaryOwners` bereits vorhanden und ein Listen-Datentyp sein muss, damit diese Anweisung gültig ist. Andernfalls ist das Schlüsselwort `AT` in der `INSERT INTO`-Klausel erforderlich.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
```

```
WHERE r.VIN = '1N4AL11D75C109151'
```

Fügen Sie `{'PersonId':'abc123'}` als erstes Element in der vorhandenen `Owners.SecondaryOwners`-Liste innerhalb eines Dokuments ein.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
WHERE r.VIN = '1N4AL11D75C109151'
```

Hängen Sie mehrere Name-Wert-Paare an die bestehende `Owners.SecondaryOwners`-Liste innerhalb eines Dokuments an.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
WHERE r.VIN = '1N4AL11D75C109151'
```

Entfernen Sie ein vorhandenes Element in einem Dokument.

```
UPDATE Person AS p
REMOVE p.Address
WHERE p.GovId = '111-22-3333'
```

Entfernen Sie ein ganzes Dokument aus einer Tabelle.

```
UPDATE Person AS p
REMOVE p
WHERE p.GovId = '111-22-3333'
```

Entfernen Sie das erste Element in der `Owners.SecondaryOwners`-Liste innerhalb eines Dokuments in der `VehicleRegistration`-Tabelle.

```
UPDATE VehicleRegistration AS r
REMOVE r.Owners.SecondaryOwners[0]
WHERE r.VIN = '1N4AL11D75C109151'
```

Programmgesteuertes Ausführen mit dem Treiber

Informationen zum programmgesteuerten Ausführen dieser Anweisung mithilfe des QLDB-Treibers finden Sie in den folgenden Tutorials unter Erste Schritte mit dem Treiber:

- Java:ErstErstErstErstErste-Erst |Kochbuchreferenz
- .NET:Erste-Schrit-T |Kochbuchreferenz
- Gehe:Schnellstart-Tutorial |Kochbuchreferenz
- Node.js:Erste-Schrit-T-T-TSchrit |Kochbuchreferenz — Referenz für
- Python:Erste Schritte |Kochbuchreferenz

Befehl UNDROP TABLE in Amazon QLDB

Verwenden Sie in Amazon QLDB den`UNDROP TABLE` Befehl, um eine Tabelle zu reaktivieren, die Sie zuvor gelöscht (d. h. deaktiviert) haben. Das Deaktivieren oder Reaktivieren einer Tabelle hat keine Auswirkungen auf ihre Dokumente oder Indizes.

> **Note**
>
> Informationen zur Zugriffskontrolle zur Ausführung dieses PartiQL-Befehls für bestimmte Tabellen finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

Themen

- Syntax
- Parameter
- Rückgabewert
- Beispiele

Syntax

```
UNDROP TABLE "tableId"
```

Parameter

„*Tabellen-ID*“

Die eindeutige ID der Tabelle, die reaktiviert werden soll, gekennzeichnet durch doppelte Anführungszeichen.

Die Tabelle muss zuvor mit DROP gelöscht worden sein, da sie dann in der Systemkatalogtabelle `information_schema.user_tables` existiert und den Status `INACTIVE` hat. Es darf auch keine aktive, vorhandene Tabelle mit demselben Namen vorhanden sein.

Rückgabewert

`tableId`— Die eindeutige ID der Tabelle, die Sie reaktiviert haben.

Beispiele

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

PartiQL-Funktionen in Amazon QLDB

PartiQL in Amazon QLDB unterstützt die folgenden integrierten Varianten von SQL-Standardfunktionen.

> **Note**
>
> Alle SQL-Funktionen, die nicht in dieser Liste enthalten sind, werden derzeit in QLDB nicht unterstützt.
> Diese Funktionsreferenz basiert auf der PartiQL-Dokumentation Integrierte Funktionen.

Unbekannter Typ (Null und fehlende) Propagierung

Sofern nicht anders angegeben, propagieren alle diese Funktionen Null- und fehlende Argumentwerte. Propagierung von `NULL` oder `MISSING` wird als Rückgabe von `NULL` definiert, wenn ein Funktionsargument entweder `NULL` oder `MISSING` ist. Es folgen Beispiele für diese Propagierung.

```
CHAR_LENGTH(null)    -- null
CHAR_LENGTH(missing) -- null (also returns null)
```

Aggregationsfunktionen

- AVG

- COUNT
- MAX
- MIN
- SIZE
- SUM

Konditionale Funktionen

- COALESCE
- EXISTS
- NULLIF

Datums- und Zeitfunktionen

- DATE_ADD
- DATE_DIFF
- EXTRACT
- UTCNOW

Skalare Funktionen

- TXID

Zeichenfolgenfunktionen

- CHAR_LENGTH
- CHARACTER_LENGTH
- LOWER
- SUBSTRING
- TRIM
- UPPER

Funktionen für die Datentypformatierung

- CAST
- TO_STRING
- TO_TIMESTAMP

AVG-Funktion in Amazon QLDB

Verwenden Sie dieAVG Funktion in Amazon QLDB, um den Durchschnitt (das arithmetische Mittel) der Eingabeausdruckwerte zurück. Diese Funktion ist mit numerischen Werten kompatibel und ignoriert NULL-Werte.

Syntax

```
AVG ( expression )
```

Argumente

`expression`

Der Feldname oder der Ausdruck eines numerischen Datentyps, auf dem die Funktion arbeitet.

Datentypen

Unterstützte Argumenttypen:

- `int`
- `decimal`
- `float`

Typ der Rückgabe: `decimal`

Beispiele

```
SELECT AVG(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 147.19
SELECT AVG(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 2.
```

Verbundene Funktionen

- COUNT
- MAX
- MIN
- SIZE
- SUM

CAST-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB dieCAST Funktion, um einen bestimmten Ausdruck in einen Wert auszuwerten und den Wert in einen bestimmten Zieldatentyp zu konvertieren. Wenn die Konvertierung nicht durchgeführt werden kann, gibt die Funktion einen

Syntax

```
CAST ( expression AS type )
```

Argumente

expression

Der Feldname oder der Ausdruck, der zu einem Wert ausgewertet wird, den die Funktion umwandelt. Die Konvertierung von Null-Werten gibt Null-Werte zurück. Dieser Parameter kann jeder der unterstützten Datentypen sein.

Typ

Der Name des Zieldatentyps für die Umwandlung. Bei diesem Parameter kann es sich um einen der unterstützten Datentypen handeln.

Rückgabetyp

Der Datentyp, der durch das Argument *type (Typ)* angegeben wird.

Beispiele

Die folgenden Beispiele zeigen die Propagierung unbekannter Typen (`NULL` oder `MISSING`).

```
CAST(null    AS null)    -- null
CAST(missing AS null)    -- null
CAST(missing AS missing) -- missing
CAST(null    AS missing) -- missing
CAST(null    AS boolean) -- null (null AS any data type name results in null)
CAST(missing AS boolean) -- missing (missing AS any data type name results in missing)
```

Jeder Wert, der kein unbekannter Typ ist, kann nicht in`NULL` oder umgewandelt werden`MISSING`.

```
CAST(true AS null)    -- error
CAST(true AS missing) -- error
CAST(1    AS null)    -- error
CAST(1    AS missing) -- error
```

In den nachstehenden Beispielen wird die Umwandlung von `AS boolean` veranschaulicht.

```
CAST(true      AS boolean) -- true no-op
CAST(0         AS boolean) -- false
CAST(1         AS boolean) -- true
CAST(`1e0`     AS boolean) -- true (float)
CAST(`1d0`     AS boolean) -- true (decimal)
CAST('a'       AS boolean) -- false
CAST('true'    AS boolean) -- true (SqlName string 'true')
CAST(``'true'`` AS boolean) -- true (Ion symbol ``'true'``)
CAST(``'false'`` AS boolean) -- false (Ion symbol ``'false'``)
```

In den nachstehenden Beispielen wird die Umwandlung von `AS integer` veranschaulicht.

```
CAST(true   AS integer) -- 1
CAST(false  AS integer) -- 0
CAST(1      AS integer) -- 1
CAST(`1d0`  AS integer) -- 1
CAST(`1d3`  AS integer) -- 1000
CAST(1.00   AS integer) -- 1
CAST(1.45   AS integer) -- 1
CAST(1.75   AS integer) -- 1
CAST('12'   AS integer) -- 12
CAST('aa'   AS integer) -- error
CAST(``'22'`` AS integer) -- 22
CAST(``'x'``  AS integer) -- error
```

In den nachstehenden Beispielen wird die Umwandlung von `AS float` veranschaulicht.

```
CAST(true   AS float) -- 1e0
CAST(false  AS float) -- 0e0
CAST(1      AS float) -- 1e0
CAST(`1d0`  AS float) -- 1e0
CAST(`1d3`  AS float) -- 1000e0
CAST(1.00   AS float) -- 1e0
CAST('12'   AS float) -- 12e0
CAST('aa'   AS float) -- error
CAST(`''22''` AS float) -- 22e0
CAST(`''x''`  AS float) -- error
```

In den nachstehenden Beispielen wird die Umwandlung von `AS decimal` veranschaulicht.

```
CAST(true   AS decimal) -- 1.
CAST(false  AS decimal) -- 0.
CAST(1      AS decimal) -- 1.
CAST(`1d0`  AS decimal) -- 1. (REPL printer serialized to 1.)
CAST(`1d3`  AS decimal) -- 1d3
CAST(1.00   AS decimal) -- 1.00
CAST('12'   AS decimal) -- 12.
CAST('aa'   AS decimal) -- error
CAST(`''22''` AS decimal) -- 22.
CAST(`''x''`  AS decimal) -- error
```

In den nachstehenden Beispielen wird die Umwandlung von `AS timestamp` veranschaulicht.

```
CAST(`2001T`                      AS timestamp) -- 2001T
CAST('2001-01-01T'                AS timestamp) -- 2001-01-01T
CAST(`''2010-01-01T00:00:00.000Z''` AS timestamp) -- 2010-01-01T00:00:00.000Z
CAST(true                         AS timestamp) -- error
CAST(2001                         AS timestamp) -- error
```

In den nachstehenden Beispielen wird die Umwandlung von `AS symbol` veranschaulicht.

```
CAST(`''xx''`                   AS symbol) -- xx (`''xx''` is an Ion symbol)
CAST('xx'                       AS symbol) -- xx ('xx' is a string)
CAST(42                         AS symbol) -- '42'
CAST(`1e0`                      AS symbol) -- '1.0'
CAST(`1d0`                      AS symbol) -- '1'
```

```
CAST(true                       AS symbol) -- 'true'
CAST(false                      AS symbol) -- 'false'
CAST(`2001T`                    AS symbol) -- '2001T'
CAST(`2001-01-01T00:00:00.000Z` AS symbol) -- '2001-01-01T00:00:00.000Z'
```

In den nachstehenden Beispielen wird die Umwandlung von `AS string` veranschaulicht.

```
CAST(``'xx'``                   AS string) -- "xx" (``'xx'`` is an Ion symbol)
CAST('xx'                       AS string) -- "xx" ('xx' is a string)
CAST(42                         AS string) -- "42"
CAST(`1e0`                      AS string) -- "1.0"
CAST(`1d0`                      AS string) -- "1"
CAST(true                       AS string) -- "true"
CAST(false                      AS string) -- "false"
CAST(`2001T`                    AS string) -- "2001T"
CAST(`2001-01-01T00:00:00.000Z` AS string) -- "2001-01-01T00:00:00.000Z"
```

In den nachstehenden Beispielen wird die Umwandlung von `AS struct` veranschaulicht.

```
CAST(`{ a: 1 }` AS struct) -- {a:1}
CAST(true       AS struct) -- err
```

In den nachstehenden Beispielen wird die Umwandlung von `AS list` veranschaulicht.

```
CAST(`[1, 2, 3]`        AS list) -- [1,2,3]
CAST(<<'a', { 'b':2 }>> AS list) -- ["a",{'b':2}]
CAST({ 'b':2 }          AS list) -- error
```

Die folgenden Beispiele sind ausführbare Anweisungen, die einige der vorherigen Beispiele enthalten.

```
SELECT CAST(true AS integer) FROM << 0 >>            -- 1
SELECT CAST('2001-01-01T' AS timestamp) FROM << 0 >> -- 2001-01-01T
SELECT CAST('xx' AS symbol) FROM << 0 >>             -- xx
SELECT CAST(42 AS string) FROM << 0 >>               -- "42"
```

Verbundene Funktionen

- TO_STRING
- TO_TIMESTAMP

CHAR_LENGTH-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB dieCHAR_LENGTH Funktion, um die Anzahl der Zeichen in der angegebenen Zeichenfolge zurückzugeben, wobei das Zeichen als einzelner Unicode-Codepunkt definiert ist.

Syntax

```
CHAR_LENGTH ( string )
```

CHAR_LENGTH ist ein Synonym für CHARACTER_LENGTH-Funktion in Amazon QLDB.

Argumente

string

Der Feldname oder Ausdruck des Datentyps `string`, den die Funktion auswertet.

Rückgabetyp

`int`

Beispiele

```
SELECT CHAR_LENGTH('') FROM << 0 >>        -- 0
SELECT CHAR_LENGTH('abcdefg') FROM << 0 >> -- 7
SELECT CHAR_LENGTH('e#') FROM << 0 >>      -- 2 (because 'e#' is two code points: the
 letter 'e' and combining character U+032B)
```

Verbundene Funktionen

- LOWER
- SUBSTRING
- TRIM
- UPPER

CHARACTER_LENGTH-Funktion in Amazon QLDB

Synonym mit der Funktion CHAR_LENGTH.

Siehe CHAR_LENGTH-Funktion in Amazon QLDB.

COALESCE-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB anhand einer Liste mit einem oder mehreren Argumenten die`COALESCE` Funktion, um die Argumente in der Reihenfolge von links nach rechts auszuwerten und den ersten Wert zurückzugeben, der kein unbekannter Typ ist (`NULL`oder`MISSING`). Wenn alle Argumenttypen unbekannt sind, ist das Ergebnis `NULL`.

Die`COALESCE` Funktion verbreitet sich nicht`NULL` und`MISSING`.

Syntax

```
COALESCE ( expression [, expression, ... ] )
```

Argumente

`expression`

Die Liste der Feldnamen oder Ausdrücke, die von der Funktion ausgewertet werden. Jedes Argument kann ein beliebiges der unterstützten Datentypen sein.

Rückgabetyp

Beliebiger unterstützter Datentyp. Der Rückgabetyp ist entweder `NULL` oder entspricht dem Typ des ersten Ausdrucks, der zu einem Nicht-Null- und nicht fehlenden Wert ausgewertet wird.

Beispiele

```
SELECT COALESCE(1, null) FROM << 0 >>          -- 1
SELECT COALESCE(null, null, 1) FROM << 0 >>    -- 1
SELECT COALESCE(null, 'string') FROM << 0 >>  -- "string"
```

Verbundene Funktionen

- EXISTS
- NULLIF

COUNT-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB die`COUNT` Funktion, um die Anzahl der Dokumente zurückzugeben, die durch den angegebenen Ausdruck definiert sind. Diese Funktion hat zwei Varianten:

- `COUNT(*)`— Zählt alle Dokumente in der Zieltabelle, unabhängig davon, ob sie Nullwerte oder fehlende Werte enthalten.
- `COUNT(expression)`— Berechnet die Anzahl der Dokumente mit Werten ungleich Null in einem bestimmten, vorhandenen Feld oder Ausdruck.

> **Warning**
>
> Die`COUNT` Funktion ist nicht optimiert, daher empfehlen wir nicht, sie ohne eine indizierte Suche zu verwenden. Wenn Sie eine Abfrage in QLDB ohne indizierte Suche ausführen, wird ein vollständiger Tabellenscan aufgerufen. Dies kann bei großen Tabellen zu Leistungsproblemen führen, einschließlich Parallelitätskonflikten und Transaktions-Timeouts. Um Tabellenscans zu vermeiden, müssen Sie Anweisungen mit einer`WHERE` Prädikatklausel unter Verwendung eines Gleichheitsoperators (`=`oder`IN`) für ein indiziertes Feld oder eine Dokument-ID ausführen. Weitere Informationen finden Sie unter Optimieren der Abfrageleistung.

Syntax

```
COUNT ( * | expression )
```

Argumente

`expression`

Der Feldname oder der Ausdruck, auf dem die Funktion arbeitet. Dieser Parameter kann jeder der unterstützten Datentypen sein.

Rückgabetyp

`int`

Beispiele

```
SELECT COUNT(*) FROM VehicleRegistration r WHERE r.LicensePlateNumber = 'CA762X' -- 1
SELECT COUNT(r.VIN) FROM Vehicle r WHERE r.VIN = '1N4AL11D75C109151'              -- 1
SELECT COUNT(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>                   -- 3
```

Verbundene Funktionen

- AVG
- MAX
- MIN
- SIZE
- SUM

DATE_ADD-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB die`DATE_ADD` Funktion, um einen bestimmten Zeitstempelwert um ein bestimmtes Intervall zu erhöhen.

Syntax

```
DATE_ADD( datetimepart, interval, timestamp )
```

Argumente

`Datum/Uhrzeit/Teil`

Der Datums- oder Uhrzeitteil, der von der von der von der Funktion verwendet wird. Bei diesem Parameter kann es sich um einen der Folgenden handeln:

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`

Intervall

Die Ganzzahl, die das Intervall angibt, das dem angegebenen *Zeitstempel* hinzugefügt werden soll. Bei einer negativen Ganzzahl wird das Intervall subtrahiert.

timestamp

Der Feldname oder Ausdruck des Datentyps `timestamp`, den die Funktion erhöht.

Ein Ion-Timestamp-Literalwert kann mit backticks (`` `...` ``) gekennzeichnet werden. Formatierungsdetails und Beispiele für Zeitstempelwerte finden Sie unter Zeitstempel im Amazon Ion-Spezifikationsdokument.

Rückgabetyp

`timestamp`

Beispiele

```
DATE_ADD(year, 5, `2010-01-01T`)                -- 2015-01-01T
DATE_ADD(month, 1, `2010T`)                     -- 2010-02T (result adds precision as
 necessary)
DATE_ADD(month, 13, `2010T`)                    -- 2011-02T (2010T is equivalent to
 2010-01-01T00:00:00.000Z)
DATE_ADD(day, -1, `2017-01-10T`)                -- 2017-01-09T
DATE_ADD(hour, 1, `2017T`)                      -- 2017-01-01T01:00Z
DATE_ADD(hour, 1, `2017-01-02T03:04Z`)          -- 2017-01-02T04:04Z
DATE_ADD(minute, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:05:05.006Z
DATE_ADD(second, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:04:06.006Z

-- Runnable statements
SELECT DATE_ADD(year, 5, `2010-01-01T`) FROM << 0 >> -- 2015-01-01T
SELECT DATE_ADD(day, -1, `2017-01-10T`) FROM << 0 >> -- 2017-01-09T
```

Verbundene Funktionen

- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP

- UTCNOW

DATE_DIFF-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB die`DATE_DIFF` Funktion, um die Differenz zwischen den angegebenen Datumsteilen zweier gegebener Zeitstempel zurückzugeben.

Syntax

```
DATE_DIFF( datetimepart, timestamp1, timestamp2 )
```

Argumente

`Datum/Uhrzeit/Teil`

Der Datums- oder Uhrzeitteil, der von der von der von der Funktion verwendet wird. Bei diesem Parameter kann es sich um einen der Folgenden handeln:

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`

`Zeitstempel1`, *`Zeitstempel2`*

Die beiden Feldnamen oder Ausdrücke des Datentyps `timestamp`, die die Funktion vergleicht. Wenn *`Zeitstempel2`* später als *`Zeitstempel1`* ist, ist das Ergebnis positiv. Wenn *`Zeitstempel2`* früher als *`Zeitstempel1`*, ist, ist das Ergebnis negativ.

Ein Ion-Timestamp-Literalwert kann mit backticks (`` `...` ``) gekennzeichnet werden. Formatierungsdetails und Beispiele für Zeitstempelwerte finden Sie unter Zeitstempel im Amazon Ion-Spezifikationsdokument.

Rückgabetyp

`int`

Beispiele

```
DATE_DIFF(year, `2010-01-01T`, `2011-01-01T`)           -- 1
DATE_DIFF(year, `2010-12T`, `2011-01T`)                 -- 0 (must be at least 12
 months apart to evaluate as a 1 year difference)
DATE_DIFF(month, `2010T`, `2010-05T`)                   -- 4 (2010T is equivalent to
 2010-01-01T00:00:00.000Z)
DATE_DIFF(month, `2010T`, `2011T`)                      -- 12
DATE_DIFF(month, `2011T`, `2010T`)                      -- -12
DATE_DIFF(month, `2010-12-31T`, `2011-01-01T`)          -- 0 (must be at least a full
 month apart to evaluate as a 1 month difference)
DATE_DIFF(day, `2010-01-01T23:00Z`, `2010-01-02T01:00Z`) -- 0 (must be at least 24
 hours apart to evaluate as a 1 day difference)

-- Runnable statements
SELECT DATE_DIFF(year, `2010-01-01T`, `2011-01-01T`) FROM << 0 >> -- 1
SELECT DATE_DIFF(month, `2010T`, `2010-05T`) FROM << 0 >>         -- 4
```

Verbundene Funktionen

- DATE_ADD
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

EXISTS-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB bei gegebenem PartiQL-Wert die`EXISTS` Funktion, um zurückzugeben,`TRUE` ob es sich bei dem Wert um eine nicht leere Sammlung handelt. Andernfalls kehrt diese Funktion zurück`FALSE`. Wenn die Eingabe zu `EXISTS` kein Container ist, ist das Ergebnis `FALSE`.

Die`EXISTS` Funktion verbreitet sich nicht`NULL` und`MISSING`.

Syntax

```
EXISTS ( value )
```

Argumente

Wert

Der Feldname oder der Ausdruck, den die Funktion auswertet. Dieser Parameter kann jeder der unterstützten Datentypen sein.

Rückgabetyp

`bool`

Beispiele

```
EXISTS(`[]`)        -- false (empty list)
EXISTS(`[1, 2, 3]`) -- true (non-empty list)
EXISTS(`[missing]`) -- true (non-empty list)
EXISTS(`{}`)        -- false (empty struct)
EXISTS(`{ a: 1 }`)  -- true (non-empty struct)
EXISTS(`()`)        -- false (empty s-expression)
EXISTS(`(+ 1 2)`)   -- true (non-empty s-expression)
EXISTS(1)           -- false
EXISTS(`2017T`)     -- false
EXISTS(null)        -- false
EXISTS(missing)     -- error

-- Runnable statements
SELECT EXISTS(`[]`) FROM << 0 >>        -- false
SELECT EXISTS(`[1, 2, 3]`) FROM << 0 >> -- true
```

Verbundene Funktionen

- COALESCE
- NULLIF

Funktion EXTRAHIEREN in Amazon QLDB

Verwenden Sie in Amazon QLDB die`EXTRACT` Funktion, um den Ganzzahlwert eines bestimmten Datums- oder Uhrzeitteils aus einem bestimmten Zeitstempel zurückzugeben.

Syntax

```
EXTRACT ( datetimepart FROM timestamp )
```

Argumente

Datum/Uhrzeit/Teil

Der Datums- oder Uhrzeitteil, den die Funktion extrahiert. Bei diesem Parameter kann es sich um einen der Folgenden handeln:

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`
- `timezone_hour`
- `timezone_minute`

timestamp

Der Feldname oder Ausdruck des Datentyps `timestamp`, aus dem die Funktion extrahiert. Wenn dieser Parameter ein unbekannter Typ (`NULL` oder `MISSING`) ist, gibt die Funktion `NULL` zurück.

Ein Ion-Timestamp-Literalwert kann mit backticks (`` `...` ``) gekennzeichnet werden. Formatierungsdetails und Beispiele für Zeitstempelwerte finden Sie unter Zeitstempel im Amazon Ion-Spezifikationsdokument.

Rückgabetyp

`int`

Beispiele

```
EXTRACT(YEAR FROM `2010-01-01T`)                    -- 2010
EXTRACT(MONTH FROM `2010T`)                         -- 1 (equivalent to
 2010-01-01T00:00:00.000Z)
```

```
EXTRACT(MONTH FROM `2010-10T`)                              -- 10
EXTRACT(HOUR FROM `2017-01-02T03:04:05+07:08`)              -- 3
EXTRACT(MINUTE FROM `2017-01-02T03:04:05+07:08`)            -- 4
EXTRACT(TIMEZONE_HOUR FROM `2017-01-02T03:04:05+07:08`)     -- 7
EXTRACT(TIMEZONE_MINUTE FROM `2017-01-02T03:04:05+07:08`)   -- 8

-- Runnable statements
SELECT EXTRACT(YEAR FROM `2010-01-01T`) FROM << 0 >> -- 2010
SELECT EXTRACT(MONTH FROM `2010T`) FROM << 0 >>      -- 1
```

Verbundene Funktionen

- DATE_ADD
- DATE_DIFF
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

LOWER-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB die`LOWER` Funktion, um alle Großbuchstaben in einer bestimmten Zeichenfolge in Kleinbuchstaben umzuwandeln.

Syntax

```
LOWER ( string )
```

Argumente

`string`

Der Feldname oder Ausdruck des Datentyps `string`, den die Funktion umwandelt.

Rückgabetyp

`string`

Beispiele

```
SELECT LOWER('AbCdEfG!@#$') FROM << 0 >> -- 'abcdefg!@#$'
```

Verbundene Funktionen

- CHAR_LENGTH
- SUBSTRING
- TRIM
- UPPER

MAX-Funktion in Amazon QLDB

Verwenden Sie die`MAX` Funktion in Amazon QLDB, um den maximalen Wert in einem Satz von numerischen Werten zurückzugeben.

Syntax

```
MAX ( expression )
```

Argumente

`expression`

Der Feldname oder der Ausdruck eines numerischen Datentyps, auf dem die Funktion arbeitet.

Datentypen

Unterstützte Argumenttypen:

- `int`
- `decimal`
- `float`

Unterstützte Rückgabetypen:

- `int`
- `decimal`
- `float`

Beispiele

```
SELECT MAX(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 442.30
SELECT MAX(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 3
```

Verbundene Funktionen

- AVG
- COUNT
- MIN
- SIZE
- SUM

MIN-Funktion in Amazon QLDB

Verwenden Sie die`MIN` Funktion in Amazon QLDB, um den Mindestwert in einem Satz von numerischen Werten zurückzugeben.

Syntax

```
MIN ( expression )
```

Argumente

`expression`

Der Feldname oder der Ausdruck eines numerischen Datentyps, auf dem die Funktion arbeitet.

Datentypen

Unterstützte Argumenttypen:

- `int`
- `decimal`
- `float`

Unterstützte Rückgabetypen:

- `int`
- `decimal`
- `float`

Beispiele

```
SELECT MIN(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 30.45
SELECT MIN(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 1
```

Verbundene Funktionen

- AVG
- COUNT
- MAX
- SIZE
- SUM

NULLIF-Funktion in Amazon QLDB

In Amazon QLDB, die das gleiche Auswertungsergebnis haben, die das gleiche Auswertungsergebnis haben, die das`NULLIF` gleiche`NULL` Auswertungsergebnis haben, die das gleiche Auswertungsergebnis haben, die das gleiche Auswertungsergebnis haben. Andernfalls gibt diese Funktion das Auswertungsergebnis für den ersten Ausdruck zurück.

Die`NULLIF` Funktion verbreitet sich nicht`NULL` und`MISSING`.

Syntax

```
NULLIF ( expression1, expression2 )
```

Argumente

expression1, *expression2*

Die beiden Feldnamen oder Ausdrücke, die die Funktion vergleicht. Diese Parameter können jeder der unterstützten Datentypen sein.

Rückgabetyp

Beliebiger unterstützter Datentyp. Der Rückgabetyp ist NULL oder mit dem Typ des ersten Ausdrucks identisch.

Beispiele

```
NULLIF(1, 1)             -- null
NULLIF(1, 2)             -- 1
NULLIF(1.0, 1)           -- null
NULLIF(1, '1')           -- 1
NULLIF([1], [1])         -- null
NULLIF(1, NULL)          -- 1
NULLIF(NULL, 1)          -- null
NULLIF(null, null)       -- null
NULLIF(missing, null)    -- null
NULLIF(missing, missing) -- null

-- Runnable statements
SELECT NULLIF(1, 1) FROM << 0 >>   -- null
SELECT NULLIF(1, '1') FROM << 0 >> -- 1
```

Verbundene Funktionen

- COALESCE
- EXISTS

GRÖSSEN-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB dieSIZE Funktion, um die Anzahl der Elemente in einem bestimmten Container-Datentyp (Liste, Struktur oder Tasche) zurückzugeben.

Syntax

```
SIZE ( container )
```

Argumente

Container

Der Containerfeldname oder -ausdruck, auf dem die Funktion arbeitet.

Datentypen

Unterstützte Argumenttypen:

- auflisten
- Struktur
- Tasche

Typ der Rückgabe: `int`

Wenn die Eingabe zu `SIZE` kein Container ist, löst die Funktion einen Fehler aus.

Beispiele

```
SIZE(`[]`)                  -- 0
SIZE(`[null]`)              -- 1
SIZE(`[1,2,3]`)             -- 3
SIZE(<<'foo', 'bar'>>)      -- 2
SIZE(`{foo: bar}`)          -- 1 (number of key-value pairs)
SIZE(`[{foo: 1}, {foo: 2}]`) -- 2
SIZE(12)                    -- error

-- Runnable statements
SELECT SIZE(`[]`) FROM << 0 >>      -- 0
SELECT SIZE(`[1,2,3]`) FROM << 0 >> -- 3
```

Verbundene Funktionen

- AVG

- COUNT
- MAX
- MIN
- SUM

SUBSTRING-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB die`SUBSTRING` Funktion, um eine Teilzeichenfolge aus einer bestimmten Zeichenfolge zurückzugeben. Die Teilzeichenfolge beginnt mit dem angegebenen Startindex und endet mit dem letzten Zeichen der Zeichenfolge oder bei der angegebenen Länge.

Syntax

```
SUBSTRING ( string, start-index [, length ] )
```

Argumente

`string`

Der Feldname oder Ausdruck des Datentyps `string`, aus dem eine Teilzeichenfolge extrahiert werden soll.

`start-index`

Die Startposition innerhalb der *`Zeichenfolge`*, von der aus die Extraktion gestartet werden soll. Diese Zahl kann negativ sein.

Das erste Zeichen der *`Zeichenfolge`* hat den Index 1.

`length`

(Optional) Die Anzahl der Zeichen (Codepunkte), die aus der *`Zeichenfolge`* extrahiert werden sollen, beginnt am *`Start-Index`* und endet bei (*`Start-Index`* + *`Länge`*) - 1. Mit anderen Worten, die Länge der Teilzeichenfolge. Diese Zahl darf nicht negativ sein.

Wenn dieser Parameter nicht angegeben wird, wird die Funktion bis zum Ende der *`Zeichenfolge`* fortgesetzt.

Rückgabetyp

`string`

Beispiele

```
SUBSTRING('123456789', 0)      -- '123456789'
SUBSTRING('123456789', 1)      -- '123456789'
SUBSTRING('123456789', 2)      -- '23456789'
SUBSTRING('123456789', -4)     -- '123456789'
SUBSTRING('123456789', 0, 999) -- '123456789'
SUBSTRING('123456789', 0, 2)   -- '1'
SUBSTRING('123456789', 1, 999) -- '123456789'
SUBSTRING('123456789', 1, 2)   -- '12'
SUBSTRING('1', 1, 0)           -- ''
SUBSTRING('1', 1, 0)           -- ''
SUBSTRING('1', -4, 0)          -- ''
SUBSTRING('1234', 10, 10)      -- ''

-- Runnable statements
SELECT SUBSTRING('123456789', 1) FROM << 0 >>    -- "123456789"
SELECT SUBSTRING('123456789', 1, 2) FROM << 0 >> -- "12"
```

Verbundene Funktionen

- CHAR_LENGTH
- LOWER
- TRIM
- UPPER

SUM-Funktion in Amazon QLDB

Verwenden Sie die`SUM` Funktion in Amazon QLDB, um die Summe der Eingabefeld- oder Ausdruckwerte zurück. Diese Funktion ist mit numerischen Werten kompatibel und ignoriert NULL-Werte.

Syntax

```
SUM ( expression )
```

Argumente

expression

Der Feldname oder der Ausdruck eines numerischen Datentyps, auf dem die Funktion arbeitet.

Datentypen

Unterstützte Argumenttypen:

- `int`
- `decimal`
- `float`

Unterstützte Rückgabetypen:

- `int`— Für Integer-Argumente
- `decimal`— Für Dezimal- oder Fließkommaargumente

Beispiele

```
SELECT SUM(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 735.95
SELECT SUM(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 6
```

Verbundene Funktionen

- AVG
- COUNT
- MAX
- MIN
- SIZE

TO_STRING-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB die`TO_STRING` Funktion, um eine Zeichenfolgendarstellung eines bestimmten Zeitstempels im angegebenen Formatmuster zurückzugeben.

Syntax

```
TO_STRING ( timestamp, 'format' )
```

Argumente

timestamp

Der Feldname oder Ausdruck des Datentyps `timestamp`, den die Funktion in eine Zeichenfolge umwandelt.

Ein Ion-Timestamp-Literalwert kann mit backticks (`` `...` ``) gekennzeichnet werden. Formatierungsdetails und Beispiele für Zeitstempelwerte finden Sie unter Zeitstempel im Amazon Ion-Spezifikationsdokument.

format

Der Zeichenfolgenliteralwert, der das Formatmuster des Ergebnisses in Bezug auf seine Datumsteile angibt. Informationen zu gültigen Formaten finden Sie unter Zeitstempel-Formatzeichenwertungsergebnis.

Rückgabetyp

`string`

Beispiele

```
TO_STRING(`1969-07-20T20:18Z`, 'MMMM d, y')                    -- "July 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMM d, yyyy')                  -- "Jul 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'M-d-yy')                       -- "7-20-69"
TO_STRING(`1969-07-20T20:18Z`, 'MM-d-y')                       -- "07-20-1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMMM d, y h:m a')              -- "July 20, 1969 8:18
 PM"
TO_STRING(`1969-07-20T20:18Z`, 'y-MM-dd''T''H:m:ssX')          --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00Z`, 'y-MM-dd''T''H:m:ssX')    --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXX')  --
 "1969-07-20T20:18:00+0800"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXXX') --
 "1969-07-20T20:18:00+08:00"
```

```
-- Runnable statements
SELECT TO_STRING(`1969-07-20T20:18Z`, 'MMMM d, y') FROM << 0 >>           -- "July 20,
 1969"
SELECT TO_STRING(`1969-07-20T20:18Z`, 'y-MM-dd''T''H:m:ssX') FROM << 0 >> --
 "1969-07-20T20:18:00Z"
```

Verbundene Funktionen

- CAST
- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_TIMESTAMP
- UTCNOW

TO_TIMESTAMP-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB anhand einer Zeichenfolge, die einen Zeitstempel darstellt, die`TO_TIMESTAMP` Funktion, um die Zeichenfolge in einen`timestamp` Datentyp zu konvertieren. Das ist der umgekehrte Vorgang von `TO_STRING`.

Syntax

```
TO_TIMESTAMP ( string [, 'format' ] )
```

Argumente

`string`

Der Feldname oder Ausdruck des Datentyps `string`, den die Funktion in einen Zeitstempel umwandelt.

`format`

(Optional) Ein Zeichenfolgenliteralwert, der das Format der Zeichenfolge in der *`Eingabezeichenfolge`* in Bezug auf die Datumsteile definiert. Informationen zu gültigen Formaten finden Sie unter Zeitstempel-Formatzeichenwertungsergebnis.

Wenn dieses Argument weggelassen wird, geht die Funktion davon aus, dass die *`Zeichenfolge`* im Format eines standardmäßigen Ion-Zeitstempelsvorliegt. Dies ist die empfohlene Methode, einen Ion-Zeitstempel mit dieser Funktion zu analysieren.

Auffüllung mit Nullen ist optional, wenn ein Ein-Zeichen-Formatsymbol (wie `y`, `M`, `d`, `H`, `h`, `m`, `s`) verwendet wird, aber für mit Nullen aufgefüllte Varianten (wie `yyyy`, `MM`, `dd`, `HH`, `hh`, `mm`, `ss`) erforderlich.

Zweistellige Jahreszahlen (Formatsymbolyy) werden besonders behandelt. 1900 wird zu Werten größer oder gleich 70 und 2000 zu Werten unter 70 addiert.

Monatsnamen und AM- oder PM-Indikatoren unterscheiden nicht zwischen Groß- und Kleinschreibung.

Rückgabetyp

`timestamp`

Beispiele

```
TO_TIMESTAMP('2007T')                          -- `2007T`
TO_TIMESTAMP('2007-02-23T12:14:33.079-08:00') -- `2007-02-23T12:14:33.079-08:00`
TO_TIMESTAMP('2016', 'y')                      -- `2016T`
TO_TIMESTAMP('2016', 'yyyy')                   -- `2016T`
TO_TIMESTAMP('02-2016', 'MM-yyyy')             -- `2016-02T`
TO_TIMESTAMP('Feb 2016', 'MMM yyyy')           -- `2016-02T`
TO_TIMESTAMP('February 2016', 'MMMM yyyy')     -- `2016-02T`

-- Runnable statements
SELECT TO_TIMESTAMP('2007T') FROM << 0 >>              -- 2007T
SELECT TO_TIMESTAMP('02-2016', 'MM-yyyy') FROM << 0 >> -- 2016-02T
```

Verbundene Funktionen

- CAST
- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING

- UTCNOW

TRIM-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB die`TRIM` Funktion, um eine bestimmte Zeichenfolge zu kürzen, indem Sie die führenden und nachfolgenden Leerzeichen oder einen bestimmten Zeichensatz entfernen.

Syntax

```
TRIM ( [ LEADING | TRAILING | BOTH [ characters ] FROM ] string)
```

Argumente

LEADING

(Optional) Gibt an, dass die Leerzeichen oder angegebenen Zeichen vom Anfang der *`Zeichenfolge`* entfernt werden sollen. Wenn nicht angegeben, ist das Standardverhalten `BOTH`.

TRAILING

(Optional) Gibt an, dass die Leerzeichen oder angegebenen Zeichen vom Ende der *`Zeichenfolge`* entfernt werden sollen. Wenn nicht angegeben, ist das Standardverhalten `BOTH`.

BOTH

(Optional) Gibt an, dass sowohl die führenden als auch die nachfolgenden Leerzeichen oder die angegebenen Zeichen vom Anfang und Ende der *`Zeichenfolge`* entfernt werden sollen.

`Zeichen`

(Optional) Der Satz der zu entfernenden Zeichen, angegeben als `string`.

Wenn dieser Parameter nicht angegeben wird, werden Leerzeichen entfernt.

`string`

Der Feldname oder Ausdruck des Datentyps `string`, den die Funktionen kürzt.

Rückgabetyp

`string`

Beispiele

```
TRIM('       foobar         ')               -- 'foobar'
TRIM('      \tfoobar\t         ')            -- '\tfoobar\t'
TRIM(LEADING FROM '       foobar         ')  -- 'foobar         '
TRIM(TRAILING FROM '       foobar         ') -- '       foobar'
TRIM(BOTH FROM '       foobar         ')     -- 'foobar'
TRIM(BOTH '1' FROM '11foobar11')             -- 'foobar'
TRIM(BOTH '12' FROM '1112211foobar22211122') -- 'foobar'

-- Runnable statements
SELECT TRIM('       foobar         ') FROM << 0 >>              -- "foobar"
SELECT TRIM(LEADING FROM '       foobar         ') FROM << 0 >> -- "foobar         "
```

Verbundene Funktionen

- CHAR_LENGTH
- LOWER
- SUBSTRING
- UPPER

TXID-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB die`TXID` Funktion, um die eindeutige Transaktions-ID der aktuellen Anweisung zurückzugeben, die Sie ausführen. Dies ist der Wert, der dem `txId`-Metadatenfeld eines Dokuments zugewiesen wird, wenn die aktuelle Transaktion an das Journal übergeben wird.

Syntax

```
TXID()
```

Argumente

Keine

Rückgabetyp

`string`

Beispiele

```
SELECT TXID() FROM << 0 >>  -- "L7S9iJqcn9W2M4qOEn27ay"
SELECT TXID() FROM Person WHERE GovId = 'LEWISR261LL'  -- "BKeMb48PNyvHWJGZHkaodG"
```

UPPER-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB dieUPPER Funktion, um alle Kleinbuchstaben in einer bestimmten Zeichenfolge in Großbuchstaben umzuwandeln.

Syntax

```
UPPER ( string )
```

Argumente

`string`

Der Feldname oder Ausdruck des Datentyps `string`, den die Funktion umwandelt.

Rückgabetyp

`string`

Beispiele

```
SELECT UPPER('AbCdEfG!@#$') FROM << 0 >> -- 'ABCDEFG!@#$'
```

Verbundene Funktionen

- CHAR_LENGTH
- LOWER
- SUBSTRING
- TRIM

UTCNOW-Funktion in Amazon QLDB

Verwenden Sie in Amazon QLDB die`UTCNOW` Funktion, um die aktuelle Uhrzeit in koordinierter Weltzeit (UTC) als zurückzugeben`timestamp`.

Syntax

```
UTCNOW()
```

Für diese Funktion müssen Sie eine`FROM` Klausel in einer`SELECT` Abfrage angeben.

Argumente

Keine

Rückgabetyp

`timestamp`

Beispiele

```
SELECT UTCNOW() FROM << 0 >>  -- 2019-12-27T20:12:16.999Z
SELECT UTCNOW() FROM Person WHERE GovId = 'LEWISR261LL'  -- 2019-12-27T20:12:26.999Z

INSERT INTO Person VALUE { 'firstName': 'Jane', 'createdAt': UTCNOW() }
UPDATE Person p SET p.updatedAt = UTCNOW() WHERE p.firstName = 'John'
```

Verbundene Funktionen

- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP

Zeitstempel-Formatzeichenwertungsergebnis

Dieser Abschnitt enthält Referenzinformationen für Zeitstempel-Formatzeichenfolgen.

Zeitstempel-Formatzeichenfolgen gelten für die Funktionen `TO_STRING` und `TO_TIMESTAMP`. Diese Zeichenfolgen können Datumsteiltrennzeichen (wie `-`, `/` oder `:`) und die folgenden Formatsymbole enthalten.

Format	Beispiel	Beschreibung
yy	70	Zweistelliges Jahr
y	1970	Vierstelliges Jahr
yyyy	1970	Mit Nullen aufgefülltes vierstelliges Jahr
M	1	Monat Ganzzahl
MM	01	Mit Nullen aufgefüllter Monat (Ganzzahl)
MMM	. Jan.	Abgekürzter Monatsname
MMMM	Januar	Vollständiger Monatsname
d	2	Tag des Monats (1—31)
dd	02	Mit Nullen aufgefüllt (01—31)
a	AM oder PM	Meridiananzeige (für die 12-Stunden-Uhr)
h	3	Stunde (12-Stunden-Uhr, 1—12)
hh	03	Nullen aufgefüllt (12-Stunden-Uhr, 01—12)
H	3	Stunde (24-Stunden-Uhr, 0—23)
HH	03	Nullen aufgefüllt (24-Stunden-Uhr, 00—23)

Format	Beispiel	Beschreibung
`m`	4	Minuten (0—59)
`mm`	04	Mit Null aufgefüllte Minuten (00—59)
`s`	5	Sekunden (0—59)
`ss`	05	Mit Null aufgefüllte Sekunden (00—59)
`S`	0	Sekundenbruchteil (Genauigkeit: Bereich: 0,9, Bereich: 0,9)
`SS`	06	Sekundenbruchteil (Genauigkeit: 0,99, Bereich: 0,99)
`SSS`	060	Sekundenbruchteil (Genauigkeit: 0,999, Bereich: 0,0,999)
`X`	+07 oder Z	Offset von UTC in Stunden oder „Z“ bei Offset = 0 ist
`XX`	+0700 oder Z	Offset von UTC in Stunden und Minuten oder „Z“ bei Offset = 0
`XXX`	+ 07:00 oder Z	Offset von UTC in Stunden und Minuten oder „Z“ bei Offset = 0
`x`	+07	Offset von UTC in Stunden
`xx`	+0700	Offset von UTC in Stunden und Minuten
`xxx`	+07:00	Offset von UTC in Stunden und Minuten

Speichern von PartiQL-Prozeduren in Amazon QLDB

In Amazon QLDB können Sie den`EXEC` Befehl verwenden, um gespeicherte PartiQL-Prozeduren in der folgenden Syntax auszuführen.

```
EXEC stored_procedure_name argument [, ... ]
```

QLDB unterstützt nur die folgenden gespeicherten Systemprozeduren:

Themen

- Die gespeicherte Prozedur REDACT_REVISION in Amazon QLDB

Die gespeicherte Prozedur REDACT_REVISION in Amazon QLDB

> **Note**
>
> Alle Bücher, die vor dem 22. Juli 2021 erstellt wurden, können derzeit nicht bearbeitet werden. Sie können die Erstellungszeit Ihres Ledgers in der Amazon QLDB-Konsole einsehen.

Verwenden Sie in Amazon QLDB die`REDACT_REVISION` gespeicherte Prozedur, um eine einzelne, inaktive Dokumentrevision sowohl im indexierten Speicher als auch im Journalspeicher dauerhaft zu löschen. Diese gespeicherte Prozedur löscht alle Benutzerdaten in der angegebenen Revision. Die Journalsequenz und die Metadaten des Dokuments, einschließlich der Dokument-ID und des Hashs, bleiben jedoch unverändert. Dieser Vorgang ist irreversibel.

Bei der angegebenen Dokumentrevision muss es sich um eine inaktive Version in der Historie handeln. Die letzte aktive Revision eines Dokuments kann nicht bearbeitet werden.

Nachdem Sie eine Redigierungsanfrage gesendet haben, indem Sie diese gespeicherte Prozedur ausführen, verarbeitet QLDB die Redigierung von Daten asynchron. Nach Abschluss einer Bearbeitung werden die Benutzerdaten der angegebenen Revision (dargestellt durch die`data` Struktur) durch ein neues`dataHash` Feld ersetzt. Der Wert dieses Feldes ist der Amazon Ion-Hash der entfernten`data` Struktur. Dadurch behält das Ledger seine allgemeine Datenintegrität bei und bleibt durch die vorhandenen Verifizierungs-API-Operationen kryptografisch überprüfbar.

Ein Beispiel für einen Schwärzvorgang mit Beispieldaten finden SieBeispiel für eine Redigierung unterRedigieren von Dokumentrevisionen.

> **Note**
>
> Informationen zur Zugriffskontrolle zur Ausführung dieses PartiQL-Befehls für bestimmte Tabellen finden Sie unterErste Schritte mit dem Standard-Berechtigungsmodus in Amazon QLDB.

Themen

Überlegungen und Einschränkungen

Bevor Sie mit der Datenredaktion in Amazon QLDB beginnen, stellen Sie sicher, dass Sie die folgenden Überlegungen und Einschränkungen überprüfen:

- Die`REDACT_REVISION` gespeicherte Prozedur zielt auf Ihre Benutzerdaten in einer einzelnen, inaktiven Dokumentrevision ab. Um mehrere Revisionen zu redigieren, müssen Sie die gespeicherte Prozedur für jede Revision einmal ausführen. Sie können eine Revision pro Transaktion redigieren.
- Um bestimmte Felder innerhalb einer Dokumentversion zu redigieren, müssen Sie zuerst eine separate DML-Anweisung (Data Manipulation Language) verwenden, um die Revision zu ändern. Weitere Informationen finden Sie unter Ein bestimmtes Feld innerhalb einer Revision redigieren.
- Nachdem QLDB eine Redigierungsanfrage erhalten hat, können Sie die Anfrage nicht stornieren oder ändern. Um zu überprüfen, ob eine Bearbeitung abgeschlossen ist, können Sie überprüfen, ob die`data` Struktur einer Revision durch ein`dataHash` Feld ersetzt wurde. Weitere Informationen hierzu finden Sie unter Überprüfung, ob eine Bearbeitung abgeschlossen ist.
- Die Redaktion hat keine Auswirkungen auf QLDB-Daten, die außerhalb des QLDB-Dienstes repliziert werden. Dazu gehören alle Exporte nach Amazon S3 und Streams zu Amazon Kinesis

Data Streams. Sie müssen andere Datenspeichermethoden verwenden, um alle außerhalb von QLDB gespeicherten Daten zu verwalten.

- Die Redaktion hat keinen Einfluss auf die wörtlichen Werte in PartiQL-Anweisungen, die im Journal aufgezeichnet sind. Als Best Practice sollten Sie parametrisierte Anweisungen programmgesteuert ausführen, indem Sie variable Platzhalter anstelle von Literalwerten verwenden. Ein Platzhalter wird im Journal als Fragezeichen (?) anstelle von vertraulichen Informationen, die möglicherweise geschwärzt werden müssen, geschrieben.

  Informationen zum programmgesteuerten Ausführen von PartiQL-Anweisungen mithilfe des QLDB-Treibers finden Sie in den Tutorials für jede unterstützte Programmiersprache unterErste Schritte mit dem Treiber.

Syntax

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

Argumente

`` `Blockadresse` ``

Der Journalblock, an dem sich die Revision des Dokuments befindet, das redigiert werden soll. Eine Adresse ist eine Amazon Ion-Struktur, die aus zwei Feldern besteht:`strandId` und`sequenceNo`.

Dies ist ein Ion-Literalwert, der mit Backticks bezeichnet wird. Beispiel:

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

Informationen zum Finden der Blockadresse finden Sie unterMetadaten von Dokumenten abfragen.

`'Tabellen-ID'`

Die eindeutige ID der Tabelle, deren Dokumentversion Sie schwärzen möchten, gekennzeichnet durch einfache Anführungszeichen.

Informationen zum Ermitteln der Tabellen-ID finden Sie unterAbfragen des Systemkatalogs.

'*Dokument-ID*'

Die eindeutige Dokument-ID der Revision, die redigiert werden soll, gekennzeichnet durch einfache Anführungszeichen.

Informationen zum Auffinden der Dokument-ID finden Sie unterMetadaten von Dokumenten abfragen.

Rückgabewert

Eine Amazon Ion-Struktur, die die zu redigierende Dokumentversion im folgenden Format darstellt.

```
{
  blockAddress: {
    strandId: String,
    sequenceNo: Int
  },
  tableId: String,
  documentId: String,
  version: Int
}
```

Strukturfelder zurückgeben

- `blockAddress`— Die Position des Journalblocks, in dem sich die Revision befindet, die redigiert werden soll. Eine Adresse hat die folgenden beiden Felder.
  - `strandId`— Die eindeutige ID des Journalstrangs, der den Block enthält.
  - `sequenceNo`— Eine Indexnummer, die die Position des Blocks innerhalb des Strangs angibt.
- `tableId`— Die eindeutige ID der Tabelle, deren Revision Sie redigieren.
- `documentId`— Die eindeutige Dokument-ID der Revision, die redigiert werden soll.
- `version`— Die Versionsnummer der Dokumentrevision, die redigiert werden soll.

Das folgende Beispiel ein Rückgabestruktur mit Beispieldaten.

```
{
  blockAddress: {
    strandId: "CsRnx0RDoNK6ANEEePa1ov",
    sequenceNo: 134
```

```
  },
  tableId: "6GZumdHggklLdMGyQq9DNX",
  documentId: "IXlQPSbfyKMIIsygePeKrZ",
  version: 0
}
```

Beispiele

```
EXEC REDACT_REVISION `{strandId:"7z2P0AyQKWD8oFYmGNhi8D", sequenceNo:7}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

PartiQL-Operatoren in Amazon QLDB

PartiQL in Amazon QLDB unterstützt die folgenden SQL-Standardoperatoren.

> **Note**
>
> SQL-Operatoren, die nicht in dieser Liste enthalten sind, werden derzeit in QLDB nicht unterstützt.

Arithmetische Operatoren

Operator	Beschreibung
+	Addition
-	Subtraktion
*	Multiply (Multiplikation)
/	Division
%	Modulo

Vergleichsoperatoren

Operator	Beschreibung
=	gleich
>	größer als
<	kleiner als
>=	größer als oder gleich
<=	kleiner als oder gleich
<>	nicht gleich

Logische Operatoren

Operator	Beschreibung
AND	TRUE, wenn alle durch AND getrennten Bedingungen TRUE sind
BETWEEN	TRUE, wenn der Operand innerhalb des Vergleichsbereichs liegt
IN	TRUE, wenn der Operand einer Liste von Ausdrücken gleich ist
IS	TRUE, wenn der Operand ein bestimmte r Datentyp ist, einschließlich NULL oder MISSING
LIKE	TRUE, wenn der Operand mit einem Muster übereinstimmt
NOT	Kehrt den Wert eines gegebenen booleschen Ausdrucks um

Operator	Beschreibung
OR	TRUE, wenn eine der von OR getrennten Bedingungen TRUE ist

Zeichenfolgenoperatoren

Operator	Beschreibung
||	Verkettet zwei Zeichenfolgen auf beiden Seiten des ||-Operators und gibt die verkettete Zeichenfolge zurück. Wenn eine oder beide Zeichenfolgen NULL ist/sind, ist das Ergebnis der Verkettung Null.

Reservierte Schlüsselwörter in Amazon QLDB

Die folgende Liste enthält reservierte PartiQL-Schlüsselwörter in Amazon QLDB. Sie können ein reserviertes Schlüsselwort als Bezeichner in Anführungszeichen mit doppelten Anführungszeichen verwenden (z. B.`"user"`). Hinweise zu PartiQL-Zitierkonventionen in QLDB finden Sie unterAbfragen von Ion mit PartiQL.

> **Important**
>
> Die Schlüsselwörter in dieser Liste gelten als reserviert, da PartiSQL mit SQL-92 abwärtskompatibel ist. QLDB unterstützt jedoch nur eine Teilmenge dieser reservierten Wörter. Die Liste der SQL-Schlüsselwörter, die QLDB derzeit unterstützt, finden Sie in den folgenden Themen:
>
> - PartiQL-Funktionen
> - PartiQL-Operatoren
> - PartiQL-Befehle

```
ABSOLUTE
```

```
ACTION
ADD
ALL
ALLOCATE
ALTER
AND
ANY
ARE
AS
ASC
ASSERTION
AT
AUTHORIZATION
AVG
BAG
BEGIN
BETWEEN
BIT
BIT_LENGTH
BLOB
BOOL
BOOLEAN
BOTH
BY
CASCADE
CASCADED
CASE
CAST
CATALOG
CHAR
CHARACTER
CHARACTER_LENGTH
CHAR_LENGTH
CHECK
CLOB
CLOSE
COALESCE
COLLATE
COLLATION
COLUMN
COMMIT
CONNECT
CONNECTION
CONSTRAINT
```

```
CONSTRAINTS
CONTINUE
CONVERT
CORRESPONDING
COUNT
CREATE
CROSS
CURRENT
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_USER
CURSOR
DATE
DATE_ADD
DATE_DIFF
DAY
DEALLOCATE
DEC
DECIMAL
DECLARE
DEFAULT
DEFERRABLE
DEFERRED
DELETE
DESC
DESCRIBE
DESCRIPTOR
DIAGNOSTICS
DISCONNECT
DISTINCT
DOMAIN
DOUBLE
DROP
ELSE
END
END-EXEC
ESCAPE
EXCEPT
EXCEPTION
EXEC
EXECUTE
EXISTS
EXTERNAL
```

```
EXTRACT
FALSE
FETCH
FIRST
FLOAT
FOR
FOREIGN
FOUND
FROM
FULL
GET
GLOBAL
GO
GOTO
GRANT
GROUP
HAVING
HOUR
IDENTITY
IMMEDIATE
IN
INDEX
INDICATOR
INITIALLY
INNER
INPUT
INSENSITIVE
INSERT
INT
INTEGER
INTERSECT
INTERVAL
INTO
IS
ISOLATION
JOIN
KEY
LANGUAGE
LAST
LEADING
LEFT
LEVEL
LIKE
LIMIT
```

```
LIST
LOCAL
LOWER
MATCH
MAX
MIN
MINUTE
MISSING
MODULE
MONTH
NAMES
NATIONAL
NATURAL
NCHAR
NEXT
NO
NOT
NULL
NULLIF
NUMERIC
OCTET_LENGTH
OF
ON
ONLY
OPEN
OPTION
OR
ORDER
OUTER
OUTPUT
OVERLAPS
PAD
PARTIAL
PIVOT
POSITION
PRECISION
PREPARE
PRESERVE
PRIMARY
PRIOR
PRIVILEGES
PROCEDURE
PUBLIC
READ
```

```
REAL
REFERENCES
RELATIVE
REMOVE
RESTRICT
REVOKE
RIGHT
ROLLBACK
ROWS
SCHEMA
SCROLL
SECOND
SECTION
SELECT
SESSION
SESSION_USER
SET
SEXP
SIZE
SMALLINT
SOME
SPACE
SQL
SQLCODE
SQLERROR
SQLSTATE
STRING
STRUCT
SUBSTRING
SUM
SYMBOL
SYSTEM_USER
TABLE
TEMPORARY
THEN
TIME
TIMESTAMP
TIMEZONE_HOUR
TIMEZONE_MINUTE
TO
TO_STRING
TO_TIMESTAMP
TRAILING
TRANSACTION
```

```
TRANSLATE
TRANSLATION
TRIM
TRUE
TUPLE
TXID
UNDROP
UNION
UNIQUE
UNKNOWN
UNPIVOT
UPDATE
UPPER
USAGE
USER
USING
UTCNOW
VALUE
VALUES
VARCHAR
VARYING
VIEW
WHEN
WHENEVER
WHERE
WITH
WORK
WRITE
YEAR
ZONE
```

Referenz zum Amazon Ion-Datenformat in Amazon QLDB

Amazon QLDB verwendet ein Datennotationsmodell, das Amazon Ion mit einer Teilmenge von PartiQL-Typen vereint. Dieser Abschnitt bietet eine Referenzübersicht über das Ion-Dokument-Datenformat, unabhängig von dessen Integration in PartiQL.

Ion mit PartiQL in Amazon QLDB abfragen

Die Syntax und Semantik der Abfrage von Ion-Daten mit PartiQL in QLDB finden SieAbfragen von Ion mit PartiQL in der Amazon QLDB PartiQL-Referenz.

Codebeispiele, die Ion-Daten in einem QLDB-Ledger abfragen und verarbeiten, finden Sie unterBeispiele für Amazon Ion-Code undArbeiten mit.

Themen

Was ist Amazon Ion?

Hierbei handelt es sich um ein Open-Source-, stark typisiertes, selbstbeschreibendes, hierarchisches Daten-Serialisierungsformat, das ursprünglich intern bei Amazon entwickelt wurde. Es basiert auf einem abstrakten Datenmodell, mit dem Sie sowohl strukturierte als auch unstrukturierte Daten speichern können. Es ist ein Superset von JSON, was bedeutet, dass jedes gültige JSON-Dokument auch ein gültiges Ion-Dokument ist. In diesem Handbuch wird davon ausgegangen, dass ein grundlegendes praktisches Wissen zu JSON vorhanden ist. Wenn Sie mit JSON noch nicht vertraut sind, finden Sie weitere Informationen unter Einführung in JSON.

Sie können Ion-Dokumente austauschbar entweder in Form von lesbarem Text oder in binär-kodierter Form verwenden. Wie JSON lässt sich die Textform leicht lesen und schreiben und unterstützt schnelles Prototyping. Die binäre Codierung ist kompakter und lässt sich effizient beibehalten, übertragen und analysieren. Ein Ion-Prozessor kann zwischen beiden Formaten transcodieren, um genau denselben Satz an Datenstrukturen ohne Datenverlust darzustellen. Mit dieser Funktion können Anwendungen die Verarbeitung von Daten für verschiedene Anwendungsfälle optimieren.

> **Note**
>
> Das Ion-Datenmodell basiert ausschließlich auf Werten und unterstützt keine Referenzen. Auf diese Weise kann das Datenmodell Hierarchien darstellen, die in beliebiger Tiefe verschachtelt werden können, jedoch keine gezielten Diagramme.

Ion-Spezifikation

Eine vollständige Liste der Ion-Kerndatentypen mit vollständigen Beschreibungen und Details zur Wertformatierung finden Sie im Ion-Spezifikationsdokument auf der GitHub Amazon-Website.

Um die Anwendungsentwicklung zu optimieren, stellt Amazon Ion Clientbibliotheken bereit, die Ion-Daten für Sie verarbeiten. Codebeispiele für gängige Anwendungsfälle für die Verarbeitung von Ion-Daten finden Sie im Amazon Ion Cookbook unter GitHub.

JSON-kompatibel

Ähnlich wie bei JSON verfassen Sie Amazon Ion-Dokumente mit einer Reihe von primitiven Datentypen und einer Reihe von rekursiv definierten Containertypen. Ion enthält die folgenden traditionellen JSON-Datentypen:

- `null`: Ein generischer, nicht typisierter Null-Wert (leer). Außerdem unterstützt Ion einen eindeutigen Null-Typ für jeden primitiven Typ, wie im folgenden Abschnitt beschrieben.
- `bool`: Boolesche Werte.
- `string`: Unicode-Textliterale.
- `list`: Zulässige heterogene Sammlungen von Werten.
- `struct`: Nicht zulässige Sammlungen von Schlüssel-Wert-Paaren. Wie JSON erlaubt auch `struct` mehrere Werte pro Schlüssel, aber davon wird in der Regel abgeraten.

Erweiterungen von JSON

Arten von Zahlen

Anstelle des mehrdeutigen JSON-`number`-Typs, definiert Amazon Ion Zahlen streng als einen der folgenden Typen:

- `int`: Signierte Ganzzahlen von beliebiger Größe.
- `decimal`: Dezimal-kodierte echte Zahlen von beliebiger Präzision.
- `float`: Binär-kodierte Gleitkommazahlen (64-Bit-IEEE).

Beim Analysieren von Dokumenten weist ein Ion-Prozessor Zahlentypen wie folgt zu:

- `int`: Zahlen ohne Exponent oder Dezimaltrennzeichen (z. B. `100200`).

- `decimal`: Zahlen mit einem Dezimaltrennzeichen und keinem Exponent (z. B. `0.00001`, `200.0`).
- `float`: Zahlen mit einem Exponenten, wie z. B. einer wissenschaftlichen Notation oder E-Notation (z. B. `2e0`, `3.1e-4`).

Neue Datentypen

Amazon Ion fügt die folgenden Datentypen hinzu:

- `timestamp`: Datums-/Uhrzeit-/Zeitzonen-Momente von beliebiger Präzision.
- `symbol`: Symbolische Unicode-Atome (z. B. Kennungen).
- `blob`: Binäre Daten der benutzerdefinierten Kodierung.
- `clob`: Textdaten der benutzerdefinierten Kodierung.
- `sexp`: Zulässige Sammlungen von Werten mit anwendungsdefinierter Semantik.

Nulltypen

Zusätzlich zu den allgemeinen Null-Typ, die von JSON definiert sind, unterstützt Amazon Ion einen eindeutigen Null-Typ für jeden primitiven Typ. Dies gibt einen fehlenden Wert an, während ein strenger Datentyp beibehalten wird.

```
null
null.null       // Identical to untyped null
null.bool
null.int
null.float
null.decimal
null.timestamp
null.string
null.symbol
null.blob
null.clob
null.struct
null.list
null.sexp
```

Ion-Textbeispiel

```
// Here is a struct, which is similar to a JSON object.
```

```
{
    // Field names don't always have to be quoted.
    name: "fido",

    // This is an integer.
    age: 7,

    // This is a timestamp with day precision.
    birthday: 2012-03-01T,

    // Here is a list, which is like a JSON array.
    toys: [
        // These are symbol values, which are like strings,
        // but get encoded as integers in binary.
        ball,
        rope
    ],
}
```

API-Referenzen

- ion-go
- ion-java
- ion-js
- ion-python

Amazon Ion-Codebeispiele in QLDB

Dieser Abschnitt enthält Codebeispiele, die Amazon Ion-Daten verarbeiten, indem Dokumentenwerte in einem Amazon QLDB-Ledger gelesen und geschrieben werden. Die Codebeispiele verwenden den QLDB-Treiber, um PartiQL-Anweisungen auf dem Ledger auszuführen. Diese Beispiele sind Teil der Beispielanwendung inErste Schritte mit Amazon QLDB anhand eines Beispielanwendungs-Tutorials und stehen als Open Source auf der AWS GitHub Samples-Website zur Verfügung.

Allgemeine Codebeispiele, die gängige Anwendungsfälle der Verarbeitung von Ion-Daten veranschaulichen, finden Sie im Amazon Ion Cookbook unter GitHub.

Ausführen des Codes

Der Tutorial-Code für jede Programmiersprache führt die folgenden Schritte aus:

1. Stellen Sie eine Verbindung mit dem `vehicle-registration`-Beispiel-Ledger her.
2. Erstellen Sie eine Tabelle mit dem Namen „`IonTypes`“.
3. Fügen Sie ein Dokument mit einem einzigen `Name`-Feld in die Tabelle ein.
4. Für jeden unterstützten Ion-Datentyp:
   a. Aktualisieren Sie das `Name`-Dokumentfeld mit einem Literalwert des Datentyps.
   b. Fragen Sie die Tabelle ab, um die neueste Version des Dokuments zu erhalten.
   c. Überprüfen Sie, ob der Wert „`Name`“ seine ursprünglichen Datentypeigenschaften beibehalten hat, indem Sie überprüfen, ob er mit dem erwarteten Typ übereinstimmt.
5. Entfernen Sie den`IonTypes` Tisch.

> **Note**
>
> Bevor Sie diesen Tutorial-Code ausführen, müssen Sie ein Ledger mit dem Namen „`vehicle-registration`“ erstellen.

Java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
```

```
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonBlob;
import com.amazon.ion.IonBool;
import com.amazon.ion.IonClob;
import com.amazon.ion.IonDecimal;
import com.amazon.ion.IonFloat;
import com.amazon.ion.IonInt;
import com.amazon.ion.IonList;
import com.amazon.ion.IonNull;
import com.amazon.ion.IonSexp;
import com.amazon.ion.IonString;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonSymbol;
import com.amazon.ion.IonTimestamp;
import com.amazon.ion.IonValue;
import com.amazon.ion.Timestamp;
import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import java.util.Collections;
import java.util.List;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;

/**
 * Insert all the supported Ion types into a ledger and verify that they are stored
 and can be retrieved properly, retaining
 * their original properties.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-credentials.html
 */
public class InsertIonTypes {
    public static final Logger log = LoggerFactory.getLogger(InsertIonTypes.class);
    public static final String TABLE_NAME = "IonTypes";

    private InsertIonTypes() {}
```

```
   /**
    * Update a document's Name value in the database. Then, query the value of the
Name key and verify the expected Ion type was
    * saved.
    *
    * @param txn
    *              The {@link TransactionExecutor} for statement execution.
    * @param ionValue
    *              The {@link IonValue} to set the document's Name value to.
    *
    * @throws AssertionError when no value is returned for the Name key or if the
value does not match the expected type.
    */
   public static void updateRecordAndVerifyType(final TransactionExecutor txn,
final IonValue ionValue) {
       final String updateStatement = String.format("UPDATE %s SET Name = ?",
TABLE_NAME);
       final List<IonValue> parameters = Collections.singletonList(ionValue);
       txn.execute(updateStatement, parameters);
       log.info("Updated document.");

       final String searchQuery = String.format("SELECT VALUE Name FROM %s",
TABLE_NAME);
       final Result result = txn.execute(searchQuery);

       if (result.isEmpty()) {
           throw new AssertionError("Did not find any values for the Name key.");
       }
       for (IonValue value : result) {
           if (!ionValue.getClass().isInstance(value)) {
               throw new AssertionError(String.format("The queried value, %s, is
not an instance of %s.",
                       value.getClass().toString(),
ionValue.getClass().toString()));
           }
           if (!value.getType().equals(ionValue.getType())) {
               throw new AssertionError(String.format("The queried value type, %s,
does not match %s.",
                       value.getType().toString(), ionValue.getType().toString()));
           }
       }

       log.info("Successfully verified value is instance of {} with type {}.",
ionValue.getClass().toString(),
```

```
                ionValue.getType().toString());
    }

    /**
     * Delete a table.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              The name of the table to delete.
     */
    public static void deleteTable(final TransactionExecutor txn, final String
tableName) {
        log.info("Deleting {} table...", tableName);
        final String statement = String.format("DROP TABLE %s", tableName);
        txn.execute(statement);
        log.info("{} table successfully deleted.", tableName);
    }

    public static void main(final String... args) {
        final IonBlob ionBlob = Constants.SYSTEM.newBlob("hello".getBytes());
        final IonBool ionBool = Constants.SYSTEM.newBool(true);
        final IonClob ionClob = Constants.SYSTEM.newClob("{{'This is a CLOB of
text.'}}".getBytes());
        final IonDecimal ionDecimal = Constants.SYSTEM.newDecimal(0.1);
        final IonFloat ionFloat = Constants.SYSTEM.newFloat(0.2);
        final IonInt ionInt = Constants.SYSTEM.newInt(1);
        final IonList ionList = Constants.SYSTEM.newList(new int[]{1, 2});
        final IonNull ionNull = Constants.SYSTEM.newNull();
        final IonSexp ionSexp = Constants.SYSTEM.newSexp(new int[]{2, 3});
        final IonString ionString = Constants.SYSTEM.newString("string");
        final IonStruct ionStruct = Constants.SYSTEM.newEmptyStruct();
        ionStruct.put("brand", Constants.SYSTEM.newString("ford"));
        final IonSymbol ionSymbol = Constants.SYSTEM.newSymbol("abc");
        final IonTimestamp ionTimestamp =
Constants.SYSTEM.newTimestamp(Timestamp.now());

        final IonBlob ionNullBlob = Constants.SYSTEM.newNullBlob();
        final IonBool ionNullBool = Constants.SYSTEM.newNullBool();
        final IonClob ionNullClob = Constants.SYSTEM.newNullClob();
        final IonDecimal ionNullDecimal = Constants.SYSTEM.newNullDecimal();
        final IonFloat ionNullFloat = Constants.SYSTEM.newNullFloat();
        final IonInt ionNullInt = Constants.SYSTEM.newNullInt();
        final IonList ionNullList = Constants.SYSTEM.newNullList();
```

```
        final IonSexp ionNullSexp = Constants.SYSTEM.newNullSexp();
        final IonString ionNullString = Constants.SYSTEM.newNullString();
        final IonStruct ionNullStruct = Constants.SYSTEM.newNullStruct();
        final IonSymbol ionNullSymbol = Constants.SYSTEM.newNullSymbol();
        final IonTimestamp ionNullTimestamp = Constants.SYSTEM.newNullTimestamp();


        ConnectToLedger.getDriver().execute(txn -> {
            CreateTable.createTable(txn, TABLE_NAME);
            final Document document = new
Document(Constants.SYSTEM.newString("val"));
            InsertDocument.insertDocuments(txn, TABLE_NAME,
Collections.singletonList(document));

            updateRecordAndVerifyType(txn, ionBlob);
            updateRecordAndVerifyType(txn, ionBool);
            updateRecordAndVerifyType(txn, ionClob);
            updateRecordAndVerifyType(txn, ionDecimal);
            updateRecordAndVerifyType(txn, ionFloat);
            updateRecordAndVerifyType(txn, ionInt);
            updateRecordAndVerifyType(txn, ionList);
            updateRecordAndVerifyType(txn, ionNull);
            updateRecordAndVerifyType(txn, ionSexp);
            updateRecordAndVerifyType(txn, ionString);
            updateRecordAndVerifyType(txn, ionStruct);
            updateRecordAndVerifyType(txn, ionSymbol);
            updateRecordAndVerifyType(txn, ionTimestamp);

            updateRecordAndVerifyType(txn, ionNullBlob);
            updateRecordAndVerifyType(txn, ionNullBool);
            updateRecordAndVerifyType(txn, ionNullClob);
            updateRecordAndVerifyType(txn, ionNullDecimal);
            updateRecordAndVerifyType(txn, ionNullFloat);
            updateRecordAndVerifyType(txn, ionNullInt);
            updateRecordAndVerifyType(txn, ionNullList);
            updateRecordAndVerifyType(txn, ionNullSexp);
            updateRecordAndVerifyType(txn, ionNullString);
            updateRecordAndVerifyType(txn, ionNullStruct);
            updateRecordAndVerifyType(txn, ionNullSymbol);
            updateRecordAndVerifyType(txn, ionNullTimestamp);

            deleteTable(txn, TABLE_NAME);
        });
    }
```

```
    /**
     * This class represents a simple document with a single key, Name, to use for
 the IonTypes table.
     */
    private static class Document {
        private final IonValue name;

        @JsonCreator
        private Document(@JsonProperty("Name") final IonValue name) {
            this.name = name;
        }

        @JsonProperty("Name")
        private IonValue getName() {
            return name;
        }
    }
}
```

Node.js

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
```

```
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { AssertionError } from "assert";
import { dom, IonType, IonTypes } from "ion-js";

import { insertDocument } from "./InsertDocument";
import { getQldbDriver } from "./ConnectToLedger";
import { createTable } from "./CreateTable";
import { error, log } from "./qldb/LogUtil";

const TABLE_NAME: string = "IonTypes";

/**
 * Delete a table.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName Name of the table to delete.
 * @returns Promise which fulfills with void.
 */
export async function deleteTable(txn: TransactionExecutor, tableName: string):
 Promise<void> {
    log(`Deleting ${tableName} table...`);
    const statement: string = `DROP TABLE ${tableName}`;
    await txn.execute(statement);
    log(`${tableName} table successfully deleted.`);
}

/**
 * Update a document's Name value in QLDB. Then, query the value of the Name key and
 verify the expected Ion type was
 * saved.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param parameter The IonValue to set the document's Name value to.
 * @param ionType The Ion type that the Name value should be.
 * @returns Promise which fulfills with void.
 */
async function updateRecordAndVerifyType(
    txn: TransactionExecutor,
    parameter: any,
    ionType: IonType
): Promise<void> {
    const updateStatement: string = `UPDATE ${TABLE_NAME} SET Name = ?`;
    await txn.execute(updateStatement, parameter);
```

```
    log("Updated record.");

    const searchStatement: string = `SELECT VALUE Name FROM ${TABLE_NAME}`;
    const result: Result = await txn.execute(searchStatement);

    const results: dom.Value[] = result.getResultList();

    if (0 === results.length) {
        throw new AssertionError({
            message: "Did not find any values for the Name key."
        });
    }

    results.forEach((value: dom.Value) => {
        if (value.getType().binaryTypeId !== ionType.binaryTypeId) {
            throw new AssertionError({
                message: `The queried value type, ${value.getType().name}, does not
 match expected type, ${ionType.name}.`
            });
        }
    });

    log(`Successfully verified value is of type ${ionType.name}.`);
}

/**
 * Insert all the supported Ion types into a table and verify that they are stored
 and can be retrieved properly,
 * retaining their original properties.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await createTable(txn, TABLE_NAME);
            await insertDocument(txn, TABLE_NAME, [{ "Name": "val" }]);
            await updateRecordAndVerifyType(txn, dom.load("null"), IonTypes.NULL);
            await updateRecordAndVerifyType(txn, true, IonTypes.BOOL);
            await updateRecordAndVerifyType(txn, 1, IonTypes.INT);
            await updateRecordAndVerifyType(txn, 3.2, IonTypes.FLOAT);
            await updateRecordAndVerifyType(txn, dom.load("5.5"), IonTypes.DECIMAL);
            await updateRecordAndVerifyType(txn, dom.load("2020-02-02"),
 IonTypes.TIMESTAMP);
```

```
            await updateRecordAndVerifyType(txn, dom.load("abc123"),
 IonTypes.SYMBOL);
            await updateRecordAndVerifyType(txn, dom.load("\"string\""),
 IonTypes.STRING);
            await updateRecordAndVerifyType(txn, dom.load("{{ \"clob\" }}"),
 IonTypes.CLOB);
            await updateRecordAndVerifyType(txn, dom.load("{{ blob }}"),
 IonTypes.BLOB);
            await updateRecordAndVerifyType(txn, dom.load("(1 2 3)"),
 IonTypes.SEXP);
            await updateRecordAndVerifyType(txn, dom.load("[1, 2, 3]"),
 IonTypes.LIST);
            await updateRecordAndVerifyType(txn, dom.load("{brand: ford}"),
 IonTypes.STRUCT);
            await deleteTable(txn, TABLE_NAME);
        });
    } catch (e) {
        error(`Error updating and validating Ion types: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

Python

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy, modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
```

```
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from datetime import datetime
from decimal import Decimal
from logging import basicConfig, getLogger, INFO

from amazon.ion.simple_types import IonPyBool, IonPyBytes, IonPyDecimal, IonPyDict,
 IonPyFloat, IonPyInt, IonPyList, \
    IonPyNull, IonPySymbol, IonPyText, IonPyTimestamp
from amazon.ion.simpleion import loads
from amazon.ion.symbols import SymbolToken
from amazon.ion.core import IonType

from pyqldbsamples.create_table import create_table
from pyqldbsamples.constants import Constants
from pyqldbsamples.insert_document import insert_documents
from pyqldbsamples.model.sample_data import convert_object_to_ion
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)

TABLE_NAME = 'IonTypes'


def update_record_and_verify_type(driver, parameter, ion_object, ion_type):
    """
    Update a record in the database table. Then query the value of the record and
 verify correct ion type saved.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type parameter: :py:class:`amazon.ion.simple_types.IonPyValue`
    :param parameter: The Ion value or Python native type that is convertible to Ion
 for filling in parameters of the
                      statement.

    :type
 ion_object: :py:obj:`IonPyBool`/:py:obj:`IonPyBytes`/:py:obj:`IonPyDecimal`/:py:obj:`IonPyD
```

```
 /:py:obj:`IonPyFloat`/:py:obj:`IonPyInt`/:py:obj:`IonPyList`/:py:obj:`IonPyNull`

 /:py:obj:`IonPySymbol`/:py:obj:`IonPyText`/:py:obj:`IonPyTimestamp`
    :param ion_object: The Ion object to verify against.

    :type ion_type: :py:class:`amazon.ion.core.IonType`
    :param ion_type: The Ion type to verify against.

    :raises TypeError: When queried value is not an instance of Ion type.
    """
    update_query = 'UPDATE {} SET Name = ?'.format(TABLE_NAME)
    driver.execute_lambda(lambda executor: executor.execute_statement(update_query,
 parameter))
    logger.info('Updated record.')

    search_query = 'SELECT VALUE Name FROM {}'.format(TABLE_NAME)
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(search_query))

    for c in cursor:
        if not isinstance(c, ion_object):
            raise TypeError('The queried value is not an instance of
 {}'.format(ion_object.__name__))

        if c.ion_type is not ion_type:
            raise TypeError('The queried value type does not match
 {}'.format(ion_type))

    logger.info("Successfully verified value is instance of '{}' with type
 '{}'.".format(ion_object.__name__, ion_type))
    return cursor


def delete_table(driver, table_name):
    """
    Delete a table.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type table_name: str
    :param table_name: Name of the table to delete.
```

```
    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Deleting '{}' table...".format(table_name))
    cursor = driver.execute_lambda(lambda executor: executor.execute_statement('DROP
 TABLE {}'.format(table_name)))
    logger.info("'{}' table successfully deleted.".format(table_name))
    return len(list(cursor))


def insert_and_verify_ion_types(driver):
    """
    Insert all the supported Ion types and Python values that are convertible to Ion
 into a ledger and verify that they
    are stored and can be retrieved properly, retaining their original properties.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: A QLDB Driver object.
    """
    python_bytes = str.encode('hello')
    python_bool = True
    python_float = float('0.2')
    python_decimal = Decimal('0.1')
    python_string = "string"
    python_int = 1
    python_null = None
    python_datetime = datetime(2016, 12, 20, 5, 23, 43)
    python_list = [1, 2]
    python_dict = {"brand": "Ford"}

    ion_clob = convert_object_to_ion(loads('{{"This is a CLOB of text."}}'))
    ion_blob = convert_object_to_ion(python_bytes)
    ion_bool = convert_object_to_ion(python_bool)
    ion_decimal = convert_object_to_ion(python_decimal)
    ion_float = convert_object_to_ion(python_float)
    ion_int = convert_object_to_ion(python_int)
    ion_list = convert_object_to_ion(python_list)
    ion_null = convert_object_to_ion(python_null)
    ion_sexp = convert_object_to_ion(loads('(cons 1 2)'))
    ion_string = convert_object_to_ion(python_string)
    ion_struct = convert_object_to_ion(python_dict)
    ion_symbol = convert_object_to_ion(SymbolToken(text='abc', sid=123))
    ion_timestamp = convert_object_to_ion(python_datetime)
```

```
    ion_null_clob = convert_object_to_ion(loads('null.clob'))
    ion_null_blob = convert_object_to_ion(loads('null.blob'))
    ion_null_bool = convert_object_to_ion(loads('null.bool'))
    ion_null_decimal = convert_object_to_ion(loads('null.decimal'))
    ion_null_float = convert_object_to_ion(loads('null.float'))
    ion_null_int = convert_object_to_ion(loads('null.int'))
    ion_null_list = convert_object_to_ion(loads('null.list'))
    ion_null_sexp = convert_object_to_ion(loads('null.sexp'))
    ion_null_string = convert_object_to_ion(loads('null.string'))
    ion_null_struct = convert_object_to_ion(loads('null.struct'))
    ion_null_symbol = convert_object_to_ion(loads('null.symbol'))
    ion_null_timestamp = convert_object_to_ion(loads('null.timestamp'))

    create_table(driver, TABLE_NAME)
    insert_documents(driver, TABLE_NAME, [{'Name': 'val'}])
    update_record_and_verify_type(driver, python_bytes, IonPyBytes, IonType.BLOB)
    update_record_and_verify_type(driver, python_bool, IonPyBool, IonType.BOOL)
    update_record_and_verify_type(driver, python_float, IonPyFloat, IonType.FLOAT)
    update_record_and_verify_type(driver, python_decimal, IonPyDecimal,
IonType.DECIMAL)
    update_record_and_verify_type(driver, python_string, IonPyText, IonType.STRING)
    update_record_and_verify_type(driver, python_int, IonPyInt, IonType.INT)
    update_record_and_verify_type(driver, python_null, IonPyNull, IonType.NULL)
    update_record_and_verify_type(driver, python_datetime, IonPyTimestamp,
IonType.TIMESTAMP)
    update_record_and_verify_type(driver, python_list, IonPyList, IonType.LIST)
    update_record_and_verify_type(driver, python_dict, IonPyDict, IonType.STRUCT)
    update_record_and_verify_type(driver, ion_clob, IonPyBytes, IonType.CLOB)
    update_record_and_verify_type(driver, ion_blob, IonPyBytes, IonType.BLOB)
    update_record_and_verify_type(driver, ion_bool, IonPyBool, IonType.BOOL)
    update_record_and_verify_type(driver, ion_decimal, IonPyDecimal,
IonType.DECIMAL)
    update_record_and_verify_type(driver, ion_float, IonPyFloat, IonType.FLOAT)
    update_record_and_verify_type(driver, ion_int, IonPyInt, IonType.INT)
    update_record_and_verify_type(driver, ion_list, IonPyList, IonType.LIST)
    update_record_and_verify_type(driver, ion_null, IonPyNull, IonType.NULL)
    update_record_and_verify_type(driver, ion_sexp, IonPyList, IonType.SEXP)
    update_record_and_verify_type(driver, ion_string, IonPyText, IonType.STRING)
    update_record_and_verify_type(driver, ion_struct, IonPyDict, IonType.STRUCT)
    update_record_and_verify_type(driver, ion_symbol, IonPySymbol, IonType.SYMBOL)
    update_record_and_verify_type(driver, ion_timestamp, IonPyTimestamp,
IonType.TIMESTAMP)
    update_record_and_verify_type(driver, ion_null_clob, IonPyNull, IonType.CLOB)
    update_record_and_verify_type(driver, ion_null_blob, IonPyNull, IonType.BLOB)
```

```
    update_record_and_verify_type(driver, ion_null_bool, IonPyNull, IonType.BOOL)
    update_record_and_verify_type(driver, ion_null_decimal, IonPyNull,
 IonType.DECIMAL)
    update_record_and_verify_type(driver, ion_null_float, IonPyNull, IonType.FLOAT)
    update_record_and_verify_type(driver, ion_null_int, IonPyNull, IonType.INT)
    update_record_and_verify_type(driver, ion_null_list, IonPyNull, IonType.LIST)
    update_record_and_verify_type(driver, ion_null_sexp, IonPyNull, IonType.SEXP)
    update_record_and_verify_type(driver, ion_null_string, IonPyNull,
 IonType.STRING)
    update_record_and_verify_type(driver, ion_null_struct, IonPyNull,
 IonType.STRUCT)
    update_record_and_verify_type(driver, ion_null_symbol, IonPyNull,
 IonType.SYMBOL)
    update_record_and_verify_type(driver, ion_null_timestamp, IonPyNull,
 IonType.TIMESTAMP)
    delete_table(driver, TABLE_NAME)


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Insert all the supported Ion types and Python values that are convertible to Ion
 into a ledger and verify that they
    are stored and can be retrieved properly, retaining their original properties.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            insert_and_verify_ion_types(driver)
    except Exception as e:
        logger.exception('Error updating and validating Ion types.')
        raise e


if __name__ == '__main__':
    main()
```

Amazon QLDB API-Referenz

In diesem Kapitel werden die Low-Level-API-Operationen für Amazon QLDB, auf die über HTTP, dieAWS Command Line Interface(AWS CLI), oder einAWSSDK:

- Amazon QLDB— Die QLDB-Ressourcenmanagement-API (auch bekannt als dieSteuerebene) enthalten. Diese API wird nur für die Verwaltung von Ledger-Ressourcen und für nicht transaktionsbezogene Datenvorgänge verwendet. Sie können diese Operationen verwenden, um Ledger zu erstellen, zu löschen, zu beschreiben, zu beschreiben, zu beschreiben, zu beschreiben Darüber hinaus können Sie Journal-Daten kryptografisch verifizieren und Journal-Streams exportieren.
- Amazon QLDB Session— Die QLDB -Transaktionsdaten-API. Sie können diese API verwenden, um Datentransaktionen in einem Ledger mitPartiQL-Anweisungen.

> **Important**
>
> Anstatt direkt mit dem zu interagierenQLDB SessionAPI empfehlen wir die Verwendung des QLDB-Treibers oder der QLDB-Shell, um Datentransaktionen in einem Ledger auszuführen.
>
> - ArbeitenAWSSDK, verwenden Sie den QLDB-Treiber. Der Treiber stellt eine High-Level-Abstraktionsschicht überQLDB Sessiondata API und verwaltet die`SendCommand`Operation für Sie. Für weitere Informationen und eine Liste der unterstützten Programmiersprachen siehe Erste Schritte mit dem Treiber.
> - ArbeitenAWS CLIverwenden Sie die QLDB-Shell. Die -Shell ist eine Befehlszeilenschnittstelle, die den QLDB-Treiber verwendet, um mit einem LDB-Treiber zu interagieren. Weitere Informationen finden Sie unter Verwenden der Amazon QLDB-Shell (nur Daten-API).

Themen

Aktionen

Die folgenden Aktionen werden von Amazon QLDB unterstützt:

- CancelJournalKinesisStream
- CreateLedger
- DeleteLedger
- DescribeJournalKinesisStream
- DescribeJournalS3Export
- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis
- TagResource
- UntagResource
- UpdateLedger
- UpdateLedgerPermissionsMode

Die folgenden Aktionen werden von Amazon QLDB Session unterstützt:

- SendCommand

Amazon QLDB

The following actions are supported by Amazon QLDB:

- CancelJournalKinesisStream
- CreateLedger
- DeleteLedger
- DescribeJournalKinesisStream
- DescribeJournalS3Export
- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis
- TagResource
- UntagResource
- UpdateLedger
- UpdateLedgerPermissionsMode

CancelJournalKinesisStream

Service: Amazon QLDB

Beendet einen bestimmten Amazon-QLDB-Journal-Stream. Bevor ein Stream abgebrochen werden kann, muss sein aktueller Status „`ACTIVE`“ lauten.

Sie können einen Stream nicht neu starten, nachdem Sie ihn abgebrochen haben. Stornierte QLDB-Stream-Ressourcen unterliegen einem Aufbewahrungszeitraum von 7 Tagen und werden daher nach Ablauf dieses Limits automatisch gelöscht.

Anforderungssyntax

```
DELETE /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Ledgers.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

streamId

Die UUID (dargestellt in Base62-encoded Text) des QLDB-Journalstreams, der abgebrochen werden soll.

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "StreamId": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

StreamId

Die UUID (Base62-kodierter Text) des abgebrochenen QLDB-Journalstreams.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

ResourcePreconditionNotMetException

Der Vorgang ist fehlgeschlagen, da eine Bedingung nicht im Voraus erfüllt wurde.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

CreateLedger

Service: Amazon QLDB

Erstellt ein neues Hauptbuch in Ihrem AWS-Konto in der aktuellen Region.

Anforderungssyntax

```
POST /ledgers HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "Tags": {
      "string" : "string"
   }
}
```

URI-Anfrageparameter

Die Anforderung verwendet keine URI-Parameter.

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

DeletionProtection

Gibt an, ob der Ledger vor dem Löschen durch einen beliebigen Benutzer geschützt ist. Wenn es während der Ledger-Erstellung nicht definiert wird, ist diese Funktion standardmäßig aktiviert (`true`).

Wenn der Löschschutz aktiviert ist, müssen Sie ihn zuerst deaktivieren, bevor Sie das Ledger löschen können. Sie können ihn deaktivieren, indem Sie mit dem `UpdateLedger`-Vorgang den Parameter auf `false` einstellen.

Typ: Boolesch

Erforderlich: Nein

KmsKey

Der Schlüssel in AWS Key Management Service (AWS KMS) zur Verschlüsselung von Daten im Ruhezustand, die im Ledger verwendet werden. Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand im Amazon-QLDB-Entwicklerhandbuch.

Verwenden Sie eine der folgenden Optionen, um diesen Parameter anzugeben:

- `AWS_OWNED_KMS_KEY`: Einen AWS KMS-Schlüssel verwenden, der AWS gehört und in Ihrem Namen verwaltet wird.
- Undefiniert: Standardmäßig einen AWS-eigenen KMS-Schlüssel verwenden.
- Ein gültiger, symmetrischer, vom Kunden verwalteter KMS-Schlüssel: Den angegebenen symmetrischen KMS-Verschlüsselungsschlüssel in Ihrem Konto verwenden, den Sie erstellen, besitzen und verwalten.

  Amazon EBS unterstützt keine asymmetrischen Schlüssel. Weitere Informationen finden Sie unter Verwenden von symmetrischen und asymmetrischen Schlüsseln im Benutzerhandbuch für AWS Key Management Service.

Um einen vom Kunden verwalteten KMS-Schlüssel anzugeben, können Sie seine Schlüssel-ID, den Amazon-Ressourcennamen (ARN), den Aliasnamen oder den Alias-ARN verwenden. Wenn Sie einen Aliasnamen verwenden, stellen Sie ihm `"alias/"` voran. Um einen Schlüssel in einem anderen AWS-Konto anzugeben, müssen Sie seinen Schlüssel-ARN oder Alias-ARN verwenden.

Beispiel:

- Schlüssel-ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Schüssel-ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias-Name: `alias/ExampleAlias`
- Alias-ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

Weitere Informationen finden Sie unter Schlüsselkennungen (KeyId) im AWS Key Management ServiceEntwicklerhandbuch.

Typ: Zeichenfolge

Längenbeschränkungen: Die maximale Länge beträgt 1600.

Erforderlich: Nein

Name

Der Name des Ledgers, das Sie erstellen möchten. Der Name muss unter allen Ihren Ledgern in der aktuellen AWS-Konto-Region eindeutig sein.

Die Beschränkungen für die Namen von Ledgern sind im Abschnitt Kontingente in Amazon QLDB im Amazon QLDB-Entwicklerhandbuch definiert.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

PermissionsMode

Der Berechtigungsmodus, der dem Ledger zugewiesen werden soll, das Sie erstellen möchten. Folgende Parameterwerte sind möglich:

- `ALLOW_ALL`: Ein Legacy-Berechtigungsmodus, der die Zugriffskontrolle mit Granularität auf API-Ebene für Ledgers ermöglicht.

  Dieser Modus ermöglicht Benutzern, die `SendCommand`-API-Berechtigung für dieses Ledger zum Ausführen aller PartiQL-Befehle (daher `ALLOW_ALL`) auf beliebigen Tabellen im angegebenen Ledger auszuführen. Dieser Modus ignoriert alle IAM-Berechtigungs-Richtlinien auf Tabellen- oder Befehls-Ebene, die Sie für das Ledger erstellen.
- `STANDARD`: (Empfohlen) Ein Berechtigungsmodus, der Zugriffskontrolle mit feinerer Granularität für Ledger, Tabellen und PartiQL-Befehle ermöglicht.

  Standardmäßig verweigert dieser Modus alle Benutzer-Anforderungen zur Ausführung von PartiQL-Befehlen für Tabellen in diesem Ledger. Damit PartiQL-Befehle ausgeführt werden können, müssen Sie IAM-Berechtigungs-Richtlinien für bestimmte Tabellen-Ressourcen und PartiQL-Aktionen erstellen, zusätzlich zu den `SendCommand` API-Berechtigung für den Ledger. Weitere Informationen finden Sie unter Erste Schritte mit dem Standard-Berechtigungsmodus im Amazon-QLDB-Entwicklerhandbuch.

> **Note**
>
> Wir empfehlen dringend, den `STANDARD` Berechtigungsmodus, um die Sicherheit Ihrer Ledger-Daten zu maximieren.

Typ: Zeichenfolge

Zulässige Werte: `ALLOW_ALL | STANDARD`

Erforderlich: Ja

Tags

Die Schlüssel-Wert-Paare, die dem Ledger, das Sie erstellen möchten, als Tags hinzugefügt werden sollen. Bei Tag-Schlüsseln wird zwischen Groß- und Kleinschreibung unterschieden. Bei Tag-Werten wird zwischen Groß- und Kleinschreibung unterschieden und sie können Null sein.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

Karteneinträge: Mindestanzahl von 0 Elementen. Die maximale Anzahl beträgt 200 Elemente.

Schlüssel-Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 128 Zeichen.

Längenbeschränkungen für Werte: Mindestlänge von 0. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "KmsKeyArn": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "State": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

Arn

Der Amazon-Ressourcenname (ARN) für das Ledger.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

CreationDateTime

Das Datum und die Uhrzeit der Erstellung des Ledgers im Epochenzeitformat. (Das Epochenzeitformat ist die Anzahl der Sekunden, die seit dem 1. Januar 1970 um 00:00:00 Uhr UTC vergangen sind.)

Typ: Zeitstempel

DeletionProtection

Gibt an, ob der Ledger vor dem Löschen durch einen beliebigen Benutzer geschützt ist. Wenn es während der Ledger-Erstellung nicht definiert wird, ist diese Funktion standardmäßig aktiviert (`true`).

Wenn der Löschschutz aktiviert ist, müssen Sie ihn zuerst deaktivieren, bevor Sie das Ledger löschen können. Sie können ihn deaktivieren, indem Sie mit dem `UpdateLedger`-Vorgang den Parameter auf `false` einstellen.

Typ: Boolesch

KmsKeyArn

Der ARN des vom Kunden verwalteten KMS-Schlüssels, den das Ledger für die Verschlüsselung im Ruhezustand verwendet. Wenn dieser Parameter nicht definiert ist, verwendet das Ledger einen AWS eigenen KMS-Schlüssel für die Verschlüsselung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

Name

Der Name des Ledgers.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

Der Berechtigungsmodus des Ledgers, das Sie erstellt haben.

Typ: Zeichenfolge

Zulässige Werte: `ALLOW_ALL | STANDARD`

State

Der aktuelle Status des Ledgers.

Typ: Zeichenfolge

Zulässige Werte: `CREATING | ACTIVE | DELETING | DELETED`

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anfrage sind nicht gültig.

HTTP Status Code: 400

LimitExceededException

Sie haben das Limit für die maximal zulässige Anzahl von Ressourcen erreicht.

HTTP Status Code: 400

ResourceAlreadyExistsException

Die angegebene Ressource ist bereits vorhanden.

HTTP-Statuscode: 409

ResourceInUseException

Die angegebene Ressource kann derzeit nicht geändert werden.

HTTP-Statuscode: 409

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-Befehlszeilenschnittstelle
- AWS-SDK für .NET
- AWS-SDK für C++
- AWS-SDK für Go
- AWS-SDK für Java V2
- AWSSDK für JavaScript V3
- AWS-SDK für PHP V3
- AWS-SDK für Python
- AWS-SDK für Ruby V3

DeleteLedger

Service: Amazon QLDB

Löscht einen Ledger und seinen gesamten Inhalt. Diese Aktion ist unumkehrbar.

Wenn der Löschschutz aktiviert ist, müssen Sie ihn zuerst deaktivieren, bevor Sie das Ledger löschen können. Sie können ihn deaktivieren, indem Sie mit dem `UpdateLedger`-Vorgang den Parameter auf `false` einstellen.

Anforderungssyntax

```
DELETE /ledgers/name HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Ledgers, das Sie löschen möchten.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceInUseException

Die angegebene Ressource kann derzeit nicht geändert werden.

HTTP-Statuscode: 409

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

ResourcePreconditionNotMetException

Der Vorgang ist fehlgeschlagen, da eine Bedingung nicht im Voraus erfüllt wurde.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python

- AWS SDK für Ruby V3

DescribeJournalKinesisStream

Service: Amazon QLDB

Gibt detaillierte Informationen zu einem bestimmten Amazon-QLDB-Journal-Stream zurück. Die Ausgabe enthält den Amazon-Ressourcennamen (ARN), den Streamnamen, den aktuellen Status, die Erstellungszeit und die Parameter der ursprünglichen Stream-Erstellungsanforderung.

Diese Aktion gibt keine abgelaufenen Journalstreams zurück. Weitere Informationen finden Sie unter Ablauf für Terminalstreams im Amazon-QLDB-Entwicklerhandbuch.

Anforderungssyntax

```
GET /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Ledgers.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

streamId

Die UUID (dargestellt in Base62-encoded Text) des zu beschreibenden QLDB-Journal-Streams.

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Stream": {
      "Arn": "string",
      "CreationTime": number,
      "ErrorCause": "string",
      "ExclusiveEndTime": number,
      "InclusiveStartTime": number,
      "KinesisConfiguration": {
         "AggregationEnabled": boolean,
         "StreamArn": "string"
      },
      "LedgerName": "string",
      "RoleArn": "string",
      "Status": "string",
      "StreamId": "string",
      "StreamName": "string"
   }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

Stream

Informationen über den QLDB-Journal-Stream, der von einer -`DescribeJournalS3Export`Anforderung zurückgegeben wird.

Typ: JournalKinesisStreamDescription Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

ResourcePreconditionNotMetException

Der Vorgang ist fehlgeschlagen, da eine Bedingung nicht im Voraus erfüllt wurde.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

DescribeJournalS3Export

Service: Amazon QLDB

Gibt Informationen zu einem Journalexportauftrag zurück, einschließlich des Ledger-Namens, der Export-ID, der Erstellungszeit, des aktuellen Status und der Parameter der ursprünglichen Exporterstellungsanforderung.

Diese Aktion gibt keine abgelaufenen Exportaufträge zurück. Weitere Informationen finden Sie unter Ablauf von Exportaufträgen im Amazon-QLDB-Entwicklerhandbuch.

Wenn der Exportauftrag mit dem angegebenen `ExportId` nicht vorhanden ist, löst aus`ResourceNotFoundException`.

Wenn der Ledger mit dem angegebenen `Name` nicht vorhanden ist, gibt aus`ResourceNotFoundException`.

Anforderungssyntax

```
GET /ledgers/name/journal-s3-exports/exportId HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

exportId

Die UUID (dargestellt in Base62-encoded Text) des zu beschreibenden Journalexportauftrags.

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Erforderlich: Ja

name

Der Name des Ledgers.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "ExportDescription": {
      "ExclusiveEndTime": number,
      "ExportCreationTime": number,
      "ExportId": "string",
      "InclusiveStartTime": number,
      "LedgerName": "string",
      "OutputFormat": "string",
      "RoleArn": "string",
      "S3ExportConfiguration": {
         "Bucket": "string",
         "EncryptionConfiguration": {
            "KmsKeyArn": "string",
            "ObjectEncryptionType": "string"
         },
         "Prefix": "string"
      },
      "Status": "string"
   }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

ExportDescription

Informationen über den Journalexportauftrag, der von einer -`DescribeJournalS3Export`Anforderung zurückgegeben wird.

Typ: JournalS3ExportDescription Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

DescribeLedger

Service: Amazon QLDB

Gibt Informationen zu einem Ledger zurück, einschließlich Status, Berechtigungsmodus, Einstellungen für die Verschlüsselung im Ruhezustand und wann er erstellt wurde.

Anforderungssyntax

```
GET /ledgers/name HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Ledgers, das Sie beschreiben möchten.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "EncryptionDescription": {
      "EncryptionStatus": "string",
      "InaccessibleKmsKeyDateTime": number,
      "KmsKeyArn": "string"
```

```
    },
    "Name": "string",
    "PermissionsMode": "string",
    "State": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

Arn

Der Amazon-Ressourcenname (ARN) für den Ledger.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

CreationDateTime

Das Datum und die Uhrzeit im Epochenzeitformat, zu der das Ledger erstellt wurde. (Epochisches Zeitformat ist die Anzahl der Sekunden, die seit dem 1. Januar 1970 UTC um 12:00:00 Uhr verstrichen sind.)

Typ: Zeitstempel

DeletionProtection

Gibt an, ob der Ledger vor dem Löschen durch einen beliebigen Benutzer geschützt ist. Wenn es während der Ledger-Erstellung nicht definiert wird, ist diese Funktion standardmäßig aktiviert (`true`).

Wenn der Löschschutz aktiviert ist, müssen Sie ihn zuerst deaktivieren, bevor Sie das Ledger löschen können. Sie können ihn deaktivieren, indem Sie mit dem `UpdateLedger`-Vorgang den Parameter auf `false` einstellen.

Typ: Boolesch

EncryptionDescription

Informationen zur Verschlüsselung von Daten im Ruhezustand im Ledger. Dazu gehören der aktuelle Status, der AWS KMS Schlüssel und der Zeitpunkt, an dem der Schlüssel nicht mehr

zugänglich war (im Falle eines Fehlers). Wenn dieser Parameter nicht definiert ist, verwendet der Ledger einen - AWS eigenen KMS-Schlüssel für die Verschlüsselung.

Typ: LedgerEncryptionDescription Objekt

Name

Der Name des Ledgers.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

Der Berechtigungsmodus des Ledgers.

Typ: Zeichenfolge

Zulässige Werte: `ALLOW_ALL | STANDARD`

State

Der aktuelle Status des Ledgers.

Typ: Zeichenfolge

Zulässige Werte: `CREATING | ACTIVE | DELETING | DELETED`

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

ExportJournalToS3

Service: Amazon QLDB

Exportiert Journalinhalte innerhalb eines Datums- und Zeitbereichs aus einem Ledger in einen angegebenen Amazon Simple Storage Service (Amazon S3)-Bucket. Ein Journalexportauftrag kann die Datenobjekte entweder im Text- oder in der binären Darstellung des Amazon-Ion-Formats oder im JSON-Lines-Textformat schreiben.

Wenn der Ledger mit dem angegebenen `Name` nicht vorhanden ist, gibt aus`ResourceNotFoundException`.

Wenn sich der Ledger mit dem angegebenen im `CREATING` Status `Name` befindet, gibt aus`ResourcePreconditionNotMetException`.

Sie können bis zu zwei gleichzeitige Journalexportanfragen für jedes Ledger initiieren. Über dieses Limit hinaus geben Journalexportanfragen aus`LimitExceededException`.

Anforderungssyntax

```
POST /ledgers/name/journal-s3-exports HTTP/1.1
Content-type: application/json

{
   "ExclusiveEndTime": number,
   "InclusiveStartTime": number,
   "OutputFormat": "string",
   "RoleArn": "string",
   "S3ExportConfiguration": {
      "Bucket": "string",
      "EncryptionConfiguration": {
         "KmsKeyArn": "string",
         "ObjectEncryptionType": "string"
      },
      "Prefix": "string"
   }
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Ledgers.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ExclusiveEndTime

Das exklusive Enddatum und die exklusive Endzeit für den Bereich der zu exportierenden Journalinhalte.

`ExclusiveEndTime` muss im `ISO 8601`-Datums- und Uhrzeitformat sowie in UTC (Universal Coordinated Time) vorliegen. Zum Beispiel: `2019-06-13T21:36:34Z`.

`ExclusiveEndTime` muss früher oder gleich dem aktuellen UTC-Datum und der Uhrzeit sein.

Typ: Zeitstempel

Erforderlich: Ja

InclusiveStartTime

Das inklusive Startdatum und die Anfangszeit für den Bereich der zu exportierenden Journalinhalte.

`InclusiveStartTime` muss im `ISO 8601`-Datums- und Uhrzeitformat sowie in UTC (Universal Coordinated Time) vorliegen. Zum Beispiel: `2019-06-13T21:36:34Z`.

Der `InclusiveStartTime` muss vor liegen`ExclusiveEndTime`.

Wenn Sie einen angeben`InclusiveStartTime`, der vor dem des Ledgers liegt`CreationDateTime`, verwendet Amazon QLDB standardmäßig den des Ledgers`CreationDateTime`.

Typ: Zeitstempel

Erforderlich: Ja

OutputFormat

Das Ausgabeformat Ihrer exportierten Journaldaten. Ein Journalexportauftrag kann die Datenobjekte entweder im Text- oder in der binären Darstellung des Amazon-Ion-Formats oder im JSON-Lines-Textformat schreiben.

Standard: `ION_TEXT`

Im JSON Lines-Format ist jeder Journalblock in einem exportierten Datenobjekt ein gültiges JSON-Objekt, das durch eine Zeilenumbruch getrennt wird. Sie können dieses Format verwenden, um JSON-Exporte direkt in Analysetools wie Amazon Athena und AWS Glue zu integrieren, da diese Services JSON mit Zeilenumbruch automatisch analysieren können.

Typ: Zeichenfolge

Zulässige Werte: `ION_BINARY | ION_TEXT | JSON`

Erforderlich: Nein

RoleArn

Der Amazon-Ressourcenname (ARN) der IAM-Rolle, die QLDB Berechtigungen für einen Journalexportauftrag erteilt, um Folgendes zu tun:

- Schreiben Sie Objekte in Ihren Amazon S3-Bucket.
- (Optional) Verwenden Sie Ihren vom Kunden verwalteten Schlüssel in AWS Key Management Service (AWS KMS) für die serverseitige Verschlüsselung Ihrer exportierten Daten.

Um beim Anfordern eines Journalexports eine Rolle an QLDB zu übergeben, müssen Sie über Berechtigungen zum Ausführen der `iam:PassRole` Aktion für die IAM-Rollenressource verfügen. Dies ist für alle Journalexportanforderungen erforderlich.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

Erforderlich: Ja

S3ExportConfiguration

Die Konfigurationseinstellungen des Amazon S3-Bucket-Ziels für Ihre Exportanforderung.

Typ: S3ExportConfiguration Objekt

Erforderlich: Ja

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "ExportId": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

ExportId

Die UUID (dargestellt in Base62-encoded Text), die QLDB jedem Journalexportauftrag zuweist.

Um Ihre Exportanforderung zu beschreiben und den Status des Auftrags zu überprüfen, können Sie verwenden, `ExportId` um aufzurufen`DescribeJournalS3Export`.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

ResourcePreconditionNotMetException

Der Vorgang ist fehlgeschlagen, da eine Bedingung nicht im Voraus erfüllt wurde.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

GetBlock

Service: Amazon QLDB

Gibt ein Blockobjekt an einer angegebenen Adresse in einem Journal zurück. Gibt auch einen Nachweis für den angegebenen Block zur Überprüfung zurück, wenn `DigestTipAddress` bereitgestellt wird.

Informationen zu den Dateninhalten in einem Block finden Sie unter Journalinhalte im Amazon-QLDB-Entwicklerhandbuch.

Wenn der angegebene Ledger nicht vorhanden ist oder sich im `DELETING` Status befindet, löst aus`ResourceNotFoundException`.

Wenn sich der angegebene Ledger im `CREATING` Status befindet, gibt aus`ResourcePreconditionNotMetException`.

Wenn kein -Block mit der angegebenen Adresse vorhanden ist, gibt aus`InvalidParameterException`.

Anforderungssyntax

```
POST /ledgers/name/block HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Ledgers.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

BlockAddress

Der Speicherort des Blocks, den Sie anfordern möchten. Eine Adresse ist eine Amazon-Ion-Struktur mit zwei Feldern: `strandId` und `sequenceNo`.

Zum Beispiel: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`.

Typ: ValueHolder Objekt

Erforderlich: Ja

DigestTipAddress

Der letzte vom Digest abgedeckte Blockspeicherort, für den ein Nachweis angefordert werden soll. Eine Adresse ist eine Amazon-Ion-Struktur mit zwei Feldern: `strandId` und `sequenceNo`.

Zum Beispiel: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}`.

Typ: ValueHolder Objekt

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Block": {
      "IonText": "string"
   },
   "Proof": {
      "IonText": "string"
   }
```

```
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

Block

Das Blockdatenobjekt im Amazon-Ion-Format.

Typ: ValueHolder Objekt

Proof

Das Nachweisobjekt im Amazon-Ion-Format, das von einer -`GetBlock`Anforderung zurückgegeben wird. Ein Nachweis enthält die Liste der Hash-Werte, die erforderlich sind, um den angegebenen Digest mithilfe eines Merkle-Baums neu zu berechnen, beginnend mit dem angegebenen Block.

Typ: ValueHolder Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

ResourcePreconditionNotMetException

Der Vorgang ist fehlgeschlagen, da eine Bedingung nicht im Voraus erfüllt wurde.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

GetDigest

Service: Amazon QLDB

Gibt den Digest eines Ledgers am letzten festgeschriebenen Block im Journal zurück. Die Antwort enthält einen 256-Bit-Hashwert und eine Blockadresse.

Anforderungssyntax

```
POST /ledgers/name/digest HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Ledgers.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Digest": blob,
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

Digest

Der 256-Bit-Hash-Wert, der den von einer `GetDigest` Anforderung zurückgegebenen Digest darstellt.

Typ: Base64-kodiertes Binärdatenobjekt

Längenbeschränkungen: Feste Länge von 32.

DigestTipAddress

Der letzte Blockspeicherort, der vom angeforderten Digest abgedeckt wird. Eine Adresse ist eine Amazon-Ion-Struktur mit zwei Feldern: `strandId` und `sequenceNo`.

Typ: ValueHolder Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

ResourcePreconditionNotMetException

Der Vorgang ist fehlgeschlagen, da eine Bedingung nicht im Voraus erfüllt wurde.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

GetRevision

Service: Amazon QLDB

Gibt ein Revisionsdatenobjekt für eine angegebene Dokument-ID und Blockadresse zurück. Gibt auch einen Nachweis der angegebenen Revision zur Überprüfung zurück, falls `DigestTipAddress` bereitgestellt wird.

Anforderungssyntax

```
POST /ledgers/name/revision HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   },
   "DocumentId": "string"
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Ledgers.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

BlockAddress

Der Blockspeicherort der zu überprüfenden Dokumentrevision. Eine Adresse ist eine Amazon-Ion-Struktur mit zwei Feldern: `strandId` und `sequenceNo`.

Zum Beispiel: `{strandId:"BlFTjlSXze9BIh1K0szcE3",sequenceNo:14}`.

Typ: ValueHolder Objekt

Erforderlich: Ja

DigestTipAddress

Der letzte vom Digest abgedeckte Blockspeicherort, für den ein Nachweis angefordert werden soll. Eine Adresse ist eine Amazon-Ion-Struktur mit zwei Feldern: `strandId` und `sequenceNo`.

Zum Beispiel: `{strandId:"BlFTjlSXze9BIh1K0szcE3",sequenceNo:49}`.

Typ: ValueHolder Objekt

Erforderlich: Nein

DocumentId

Die UUID (dargestellt in Base62-encoded Text) des zu verifizierenden Dokuments.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Erforderlich: Ja

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Proof": {
      "IonText": "string"
   },
   "Revision": {
      "IonText": "string"
```

```
    }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

Proof

Das Nachweisobjekt im Amazon-Ion-Format, das von einer -`GetRevision`Anforderung zurückgegeben wird. Ein Nachweis enthält die Liste der Hash-Werte, die erforderlich sind, um den angegebenen Digest mithilfe eines Merkle-Baums neu zu berechnen, beginnend mit der angegebenen Dokumentrevision.

Typ: ValueHolder Objekt

Revision

Das Dokumentrevisionsdatenobjekt im Amazon-Ion-Format.

Typ: ValueHolder Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

ResourcePreconditionNotMetException

Der Vorgang ist fehlgeschlagen, da eine Bedingung nicht im Voraus erfüllt wurde.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

ListJournalKinesisStreamsForLedger

Service: Amazon QLDB

Gibt alle Amazon-QLDB-Journalstreams für einen bestimmten Ledger zurück.

Diese Aktion gibt keine abgelaufenen Journalstreams zurück. Weitere Informationen finden Sie unter Ablauf für Terminal-Streams im Amazon-QLDB-Entwicklerhandbuch.

Diese Aktion gibt maximal `MaxResults` Elemente zurück. Sie ist paginiert, sodass Sie alle Elemente abrufen können, indem Sie `ListJournalKinesisStreamsForLedger` mehrmals aufrufen.

Anforderungssyntax

```
GET /ledgers/name/journal-kinesis-streams?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Ledgers.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

MaxResults

Die maximale Anzahl der Ergebnisse, die in einer einzelnen `ListJournalKinesisStreamsForLedger` Anforderung zurückgegeben werden sollen. (Die tatsächliche Anzahl der zurückgegebenen Ergebnisse ist möglicherweise geringer.)

Gültiger Bereich: Mindestwert 1. Maximalwert 100.

NextToken

Ein Paginierungstoken, das angibt, dass Sie die nächste Seite mit Ergebnissen abrufen möchten. Wenn Sie `NextToken` in der Antwort auf einen vorherigen

`ListJournalKinesisStreamsForLedger` Aufruf einen Wert für erhalten haben, sollten Sie diesen Wert hier als Eingabe verwenden.

Längenbeschränkungen: Mindestlänge von 4. Maximale Länge beträgt 1024 Zeichen.

Pattern: `^[A-Za-z-0-9+/=]+$`

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "Streams": [
      {
         "Arn": "string",
         "CreationTime": number,
         "ErrorCause": "string",
         "ExclusiveEndTime": number,
         "InclusiveStartTime": number,
         "KinesisConfiguration": {
            "AggregationEnabled": boolean,
            "StreamArn": "string"
         },
         "LedgerName": "string",
         "RoleArn": "string",
         "Status": "string",
         "StreamId": "string",
         "StreamName": "string"
      }
   ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

NextToken

- Wenn leer `NextToken` ist, wurde die letzte Seite der Ergebnisse verarbeitet und es gibt keine Ergebnisse mehr, die abgerufen werden können.
- Wenn nicht leer `NextToken` ist, sind weitere Ergebnisse verfügbar. Um die nächste Ergebnisseite abzurufen, verwenden Sie den Wert von `NextToken` in einem nachfolgenden `ListJournalKinesisStreamsForLedger`-Aufruf.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 4. Maximale Länge beträgt 1024 Zeichen.

Pattern: `^[A-Za-z-0-9+/=]+$`

Streams

Die QLDB-Journalstreams, die derzeit dem angegebenen Ledger zugeordnet sind.

Typ: Array von JournalKinesisStreamDescription-Objekten

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

ResourcePreconditionNotMetException

Der Vorgang ist fehlgeschlagen, da eine Bedingung nicht im Voraus erfüllt wurde.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

ListJournalS3Exports

Service: Amazon QLDB

Gibt alle Journalexportaufträge für alle Ledger zurück, die dem aktuellen AWS-Konto und der Region zugeordnet sind.

Diese Aktion gibt maximal `MaxResults` Elemente zurück und wird paginiert, sodass Sie alle Elemente abrufen können, indem Sie `ListJournalS3Exports` mehrmals aufrufen.

Diese Aktion gibt keine abgelaufenen Exportaufträge zurück. Weitere Informationen finden Sie unter Ablauf von Exportaufträgen im Amazon-QLDB-Entwicklerhandbuch.

Anforderungssyntax

```
GET /journal-s3-exports?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

MaxResults

Die maximale Anzahl der Ergebnisse, die in einer einzelnen `ListJournalS3Exports` Anforderung zurückgegeben werden sollen. (Die tatsächliche Anzahl der zurückgegebenen Ergebnisse ist möglicherweise geringer.)

Gültiger Bereich: Mindestwert 1. Maximalwert 100.

NextToken

Ein Paginierungstoken, das angibt, dass Sie die nächste Seite mit Ergebnissen abrufen möchten. Wenn Sie `NextToken` in der Antwort auf einen vorherigen `ListJournalS3Exports` Aufruf einen Wert für erhalten haben, sollten Sie diesen Wert hier als Eingabe verwenden.

Längenbeschränkungen: Mindestlänge von 4. Maximale Länge beträgt 1024 Zeichen.

Pattern: `^[A-Za-z-0-9+/=]+$`

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "JournalS3Exports": [
      {
         "ExclusiveEndTime": number,
         "ExportCreationTime": number,
         "ExportId": "string",
         "InclusiveStartTime": number,
         "LedgerName": "string",
         "OutputFormat": "string",
         "RoleArn": "string",
         "S3ExportConfiguration": {
            "Bucket": "string",
            "EncryptionConfiguration": {
               "KmsKeyArn": "string",
               "ObjectEncryptionType": "string"
            },
            "Prefix": "string"
         },
         "Status": "string"
      }
   ],
   "NextToken": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

JournalS3Exports

Die Journalexportaufträge für alle Ledger, die dem aktuellen AWS-Konto und der Region zugeordnet sind.

Typ: Array von JournalS3ExportDescription-Objekten

NextToken

- Wenn leer `NextToken` ist, wurde die letzte Seite der Ergebnisse verarbeitet und es gibt keine Ergebnisse mehr, die abgerufen werden können.
- Wenn nicht leer `NextToken` ist, sind mehr Ergebnisse verfügbar. Um die nächste Ergebnisseite abzurufen, verwenden Sie den Wert von `NextToken` in einem nachfolgenden `ListJournalS3Exports`-Aufruf.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 4. Maximale Länge beträgt 1024 Zeichen.

Pattern: `^[A-Za-z-0-9+/=]+$`

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

ListJournalS3ExportsForLedger

Service: Amazon QLDB

Gibt alle Journalexportaufträge für einen angegebenen Ledger zurück.

Diese Aktion gibt maximal `MaxResults` Elemente zurück und wird paginiert, sodass Sie alle Elemente abrufen können, indem Sie `ListJournalS3ExportsForLedger` mehrmals aufrufen.

Diese Aktion gibt keine abgelaufenen Exportaufträge zurück. Weitere Informationen finden Sie unter Ablauf von Exportaufträgen im Amazon-QLDB-Entwicklerhandbuch.

Anforderungssyntax

```
GET /ledgers/name/journal-s3-exports?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

MaxResults

Die maximale Anzahl der Ergebnisse, die in einer einzelnen `ListJournalS3ExportsForLedger` Anforderung zurückgegeben werden sollen. (Die tatsächliche Anzahl der zurückgegebenen Ergebnisse ist möglicherweise geringer.)

Gültiger Bereich: Mindestwert 1. Maximalwert 100.

name

Der Name des Ledgers.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

NextToken

Ein Paginierungstoken, das angibt, dass Sie die nächste Seite mit Ergebnissen abrufen möchten. Wenn Sie `NextToken` in der Antwort auf einen vorherigen

`ListJournalS3ExportsForLedger` Aufruf einen Wert für erhalten haben, sollten Sie diesen Wert hier als Eingabe verwenden.

Längenbeschränkungen: Mindestlänge von 4. Maximale Länge beträgt 1024 Zeichen.

Pattern: `^[A-Za-z-0-9+/=]+$`

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "JournalS3Exports": [
      {
         "ExclusiveEndTime": number,
         "ExportCreationTime": number,
         "ExportId": "string",
         "InclusiveStartTime": number,
         "LedgerName": "string",
         "OutputFormat": "string",
         "RoleArn": "string",
         "S3ExportConfiguration": {
            "Bucket": "string",
            "EncryptionConfiguration": {
               "KmsKeyArn": "string",
               "ObjectEncryptionType": "string"
            },
            "Prefix": "string"
         },
         "Status": "string"
      }
   ],
   "NextToken": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

JournalS3Exports

Die Journalexportaufträge, die derzeit dem angegebenen Ledger zugeordnet sind.

Typ: Array von JournalS3ExportDescription-Objekten

NextToken

- Wenn leer `NextToken` ist, wurde die letzte Seite der Ergebnisse verarbeitet und es gibt keine Ergebnisse mehr, die abgerufen werden können.
- Wenn nicht leer `NextToken` ist, sind weitere Ergebnisse verfügbar. Um die nächste Ergebnisseite abzurufen, verwenden Sie den Wert von `NextToken` in einem nachfolgenden `ListJournalS3ExportsForLedger`-Aufruf.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 4. Maximale Länge beträgt 1024 Zeichen.

Pattern: `^[A-Za-z-0-9+/=]+$`

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python

- AWS SDK für Ruby V3

ListLedgers

Service: Amazon QLDB

Gibt alle Ledger zurück, die dem aktuellen AWS-Konto und der Region zugeordnet sind.

Diese Aktion gibt maximal `MaxResults` Elemente zurück und wird paginiert, sodass Sie alle Elemente abrufen können, indem Sie `ListLedgers` mehrmals aufrufen.

Anforderungssyntax

```
GET /ledgers?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

MaxResults

Die maximale Anzahl der Ergebnisse, die in einer einzelnen `ListLedgers` Anforderung zurückgegeben werden sollen. (Die tatsächliche Anzahl der zurückgegebenen Ergebnisse ist möglicherweise geringer.)

Gültiger Bereich: Mindestwert 1. Maximalwert 100.

NextToken

Ein Paginierungstoken, das angibt, dass Sie die nächste Seite mit Ergebnissen abrufen möchten. Wenn Sie `NextToken` in der Antwort auf einen vorherigen `ListLedgers` Aufruf einen Wert für erhalten haben, sollten Sie diesen Wert hier als Eingabe verwenden.

Längenbeschränkungen: Mindestlänge von 4. Maximale Länge beträgt 1024 Zeichen.

Pattern: `^[A-Za-z-0-9+/=]+$`

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "Ledgers": [
      {
         "CreationDateTime": number,
         "Name": "string",
         "State": "string"
      }
   ],
   "NextToken": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

Ledgers

Die Ledger, die dem aktuellen AWS-Konto und der Region zugeordnet sind.

Typ: Array von LedgerSummary-Objekten

NextToken

Ein Paginierungstoken, das angibt, ob mehr Ergebnisse verfügbar sind:

- Wenn leer `NextToken` ist, wurde die letzte Seite der Ergebnisse verarbeitet und es gibt keine Ergebnisse mehr, die abgerufen werden können.
- Wenn nicht leer `NextToken` ist, sind mehr Ergebnisse verfügbar. Um die nächste Ergebnisseite abzurufen, verwenden Sie den Wert von `NextToken` in einem nachfolgenden `ListLedgers`-Aufruf.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 4. Maximale Länge beträgt 1024 Zeichen.

Pattern: `^[A-Za-z-0-9+/=]+$`

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

ListTagsForResource

Service: Amazon QLDB

Gibt alle Tags für eine angegebene Amazon-QLDB-Ressource zurück.

Anforderungssyntax

```
GET /tags/resourceArn HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

resourceArn

Der Amazon-Ressourcenname (ARN), für den die Tags aufgelistet werden sollen. Beispielsweise:

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

Tags

Die Tags, die derzeit der angegebenen Amazon-QLDB-Ressource zugeordnet sind.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

Karteneinträge: Mindestanzahl von 0 Elementen. Die maximale Anzahl beträgt 200 Elemente.

Schlüssel-Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 128 Zeichen.

Einschränkungen der Wertlänge: Mindestlänge von 0. Maximale Länge beträgt 256 Zeichen.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2

- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

StreamJournalToKinesis

Service: Amazon QLDB

Erstellt einen Journalstream für einen bestimmten Amazon QLDB-Ledger. Der Stream erfasst jede Dokumentversion, die im Journal des Ledgers festgeschrieben wurde, und liefert die Daten an eine angegebene Amazon Kinesis Data Streams-Ressource.

Anforderungssyntax

```
POST /ledgers/name/journal-kinesis-streams HTTP/1.1
Content-type: application/json

{
   "ExclusiveEndTime": number,
   "InclusiveStartTime": number,
   "KinesisConfiguration": {
      "AggregationEnabled": boolean,
      "StreamArn": "string"
   },
   "RoleArn": "string",
   "StreamName": "string",
   "Tags": {
      "string" : "string"
   }
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Ledgers.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

ExclusiveEndTime

Das exklusive Datum und die Uhrzeit, die angeben, wann der Stream endet. Wenn Sie diesen Parameter nicht definieren, wird der Stream auf unbestimmte Zeit ausgeführt, bis Sie ihn abbrechen.

`ExclusiveEndTime` muss im `ISO 8601`-Datums- und Uhrzeitformat sowie in UTC (Universal Coordinated Time) vorliegen. Zum Beispiel: `2019-06-13T21:36:34Z`.

Typ: Zeitstempel

Erforderlich: Nein

InclusiveStartTime

Datums- und Uhrzeitangaben zum Zeitpunkt, ab dem das Streamen von Journaldaten gestartet werden soll. Diese Parameter muss im `ISO 8601`-Datums- und Uhrzeitformat sowie in UTC (Universal Coordinated Time) vorliegen. Zum Beispiel: `2019-06-13T21:36:34Z`.

`InclusiveStartTime` kann nicht in der Zukunft liegen und muss vor `ExclusiveEndTime` liegen.

Wenn Sie eine `InclusiveStartTime` angeben, die vor der `CreationDateTime` des Ledgers liegt, wird sie von QLDB standardmäßig auf die `CreationDateTime` des Ledgers festgelegt.

Typ: Zeitstempel

Erforderlich: Ja

KinesisConfiguration

Die Konfigurationseinstellungen des Ziels für Kinesis Data Streams für Ihre Stream-Anforderung.

Typ: KinesisConfiguration Objekt

Erforderlich: Ja

RoleArn

Der Amazon-Ressourcenname (ARN) der IAM-Rolle, die QLDB-Berechtigungen für einen Journalstream zum Schreiben von Datensätzen in eine Kinesis Data Streams-Ressource gewährt.

Um beim Anfordern eines Journalstreams eine Rolle an QLDB zu übergeben, müssen Sie über Berechtigungen zum Ausführen der `iam:PassRole`-Aktion für die IAM-Rollenressource verfügen. Dies ist für alle Journalstream-Anforderungen erforderlich.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

Erforderlich: Ja

StreamName

Der Name, den Sie dem QLDB-Journal-Stream zuweisen möchten. Benutzerdefinierte Namen können helfen, den Zweck eines Streams zu identifizieren und anzugeben.

Der Stream-Name muss unter anderen aktiven Streams für ein bestimmtes Ledger eindeutig sein. Stream-Namen haben dieselben Benennungsbeschränkungen wie Ledgernamen, wie sie unter Kontingente in Amazon QLDB im Amazon QLDB-Entwicklerhandbuch definiert sind.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

Tags

Die Schlüssel-Wert-Paare, die dem Stream, den Sie erstellen möchten, als Tags hinzugefügt werden sollen. Bei Tag-Schlüsseln wird zwischen Groß- und Kleinschreibung unterschieden. Bei Tag-Werten wird zwischen Groß- und Kleinschreibung unterschieden und sie können Null sein.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

Karteneinträge: Mindestanzahl von 0 Elementen. Die maximale Anzahl beträgt 200 Elemente.

Schlüssel-Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 128 Zeichen.

Einschränkungen der Wertlänge: Mindestlänge von 0. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "StreamId": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

StreamId

Die UUID (dargestellt in Base62-encoded Text), die QLDB jedem QLDB-Journal-Stream zuweist.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

ResourcePreconditionNotMetException

Der Vorgang ist fehlgeschlagen, da eine Bedingung nicht im Voraus erfüllt wurde.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

TagResource

Service: Amazon QLDB

Fügt einer angegebenen Amazon QLDB-Ressource ein oder mehrere Tags hinzu.

Eine Ressource kann bis zu 50 Tags haben. Wenn Sie versuchen, mehr als 50 Tags für eine Ressource zu erstellen, schlägt Ihre Anforderung fehl und gibt einen Fehler zurück.

Anforderungssyntax

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

resourceArn

Der Amazon-Ressourcenname (ARN), dem Sie die Tags hinzufügen möchten. Beispielsweise:

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

Tags

Die Schlüssel-Wert-Paare, die der angegebenen QLDB-Ressource als Tags hinzugefügt werden sollen. Bei Tag-Schlüsseln wird zwischen Groß- und Kleinschreibung unterschieden. Wenn Sie einen Schlüssel angeben, der bereits für die Ressource vorhanden ist, schlägt Ihre Anforderung

fehl und gibt einen Fehler zurück. Bei Tag-Werten wird zwischen Groß- und Kleinschreibung unterschieden und sie können Null sein.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

Karteneinträge: Mindestanzahl von 0 Elementen. Die maximale Anzahl beträgt 200 Elemente.

Schlüssel-Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 128 Zeichen.

Einschränkungen der Wertlänge: Mindestlänge von 0. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

Antwortsyntax

```
HTTP/1.1 200
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

UntagResource

Service: Amazon QLDB

Entfernt ein oder mehrere Tags aus einer angegebenen Amazon-QLDB-Ressource. Sie können bis zu 50 Tag-Schlüssel angeben, die entfernt werden sollen.

Anforderungssyntax

```
DELETE /tags/resourceArn?tagKeys=TagKeys HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

resourceArn

Der Amazon-Ressourcenname (ARN), aus dem die Tags entfernt werden sollen. Beispielsweise:

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

Erforderlich: Ja

TagKeys

Die Liste der zu entfernenden Tag-Schlüssel.

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP 200-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

UpdateLedger

Service: Amazon QLDB

Aktualisiert Eigenschaften in einem Ledger.

Anforderungssyntax

```
PATCH /ledgers/name HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string"
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Ledgers.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

DeletionProtection

Gibt an, ob der Ledger vor dem Löschen durch einen beliebigen Benutzer geschützt ist. Wenn es während der Ledger-Erstellung nicht definiert wird, ist diese Funktion standardmäßig aktiviert (`true`).

Wenn der Löschschutz aktiviert ist, müssen Sie ihn zuerst deaktivieren, bevor Sie das Ledger löschen können. Sie können ihn deaktivieren, indem Sie mit dem `UpdateLedger`-Vorgang den Parameter auf `false` einstellen.

Typ: Boolesch

Erforderlich: Nein

KmsKey

Der Schlüssel in AWS Key Management Service (AWS KMS), der für die Verschlüsselung von Daten im Ruhezustand im Ledger verwendet werden soll. Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand im Amazon-QLDB-Entwicklerhandbuch.

Verwenden Sie eine der folgenden Optionen, um diesen Parameter anzugeben:

- `AWS_OWNED_KMS_KEY`: Verwenden Sie einen - AWS KMS Schlüssel, der in Ihrem Namen Eigentum von ist und von AWS verwaltet wird.
- Undefiniert: Nehmen Sie keine Änderungen am KMS-Schlüssel des Ledgers vor.
- Ein gültiger, symmetrischer, vom Kunden verwalteter KMS-Schlüssel: Den angegebenen symmetrischen KMS-Verschlüsselungsschlüssel in Ihrem Konto verwenden, den Sie erstellen, besitzen und verwalten.

  Amazon EBS unterstützt keine asymmetrischen Schlüssel. Weitere Informationen finden Sie unter Verwenden von symmetrischen und asymmetrischen Schlüsseln im - AWS Key Management Service Entwicklerhandbuch.

Um einen vom Kunden verwalteten KMS-Schlüssel anzugeben, können Sie seine Schlüssel-ID, den Amazon-Ressourcennamen (ARN), den Aliasnamen oder den Alias-ARN verwenden. Wenn Sie einen Aliasnamen verwenden, stellen Sie ihm `"alias/"` voran. Um einen Schlüssel in einem anderen anzugeben AWS-Konto, müssen Sie den Schlüssel-ARN oder Alias-ARN verwenden.

Beispielsweise:

- Schlüssel-ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Schüssel-ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias-Name: `alias/ExampleAlias`
- Alias-ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

Weitere Informationen finden Sie unter Schlüsselkennungen (KeyId) im AWS Key Management Service Entwicklerhandbuch für .

Typ: Zeichenfolge

Längenbeschränkungen: Maximale Länge von 1600.

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "EncryptionDescription": {
      "EncryptionStatus": "string",
      "InaccessibleKmsKeyDateTime": number,
      "KmsKeyArn": "string"
   },
   "Name": "string",
   "State": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

Arn

Der Amazon-Ressourcenname (ARN) für den Ledger.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

CreationDateTime

Datum und Uhrzeit im Epochenzeitformat, an dem das Ledger erstellt wurde. (Epoch Time Format ist die Anzahl der Sekunden, die seit dem 12.00.00 Uhr 1. Januar 1970 UTC verstrichen sind.)

Typ: Zeitstempel

DeletionProtection

Gibt an, ob der Ledger vor dem Löschen durch einen beliebigen Benutzer geschützt ist. Wenn es während der Ledger-Erstellung nicht definiert wird, ist diese Funktion standardmäßig aktiviert (`true`).

Wenn der Löschschutz aktiviert ist, müssen Sie ihn zuerst deaktivieren, bevor Sie das Ledger löschen können. Sie können ihn deaktivieren, indem Sie mit dem `UpdateLedger`-Vorgang den Parameter auf `false` einstellen.

Typ: Boolesch

EncryptionDescription

Informationen zur Verschlüsselung von Daten im Ruhezustand im Ledger. Dazu gehören der aktuelle Status, der AWS KMS Schlüssel und der Zeitpunkt, an dem der Schlüssel unzugänglich wurde (im Falle eines Fehlers).

Typ: LedgerEncryptionDescription Objekt

Name

Der Name des Ledgers.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

State

Der aktuelle Status des Ledgers.

Typ: Zeichenfolge

Zulässige Werte: `CREATING | ACTIVE | DELETING | DELETED`

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

UpdateLedgerPermissionsMode

Service: Amazon QLDB

Aktualisiert den Berechtigungsmodus eines Ledgers.

> **Important**
>
> Bevor Sie in den `STANDARD` Berechtigungsmodus wechseln, müssen Sie zunächst alle erforderlichen IAM-Richtlinien und Tabellen-Tags erstellen, um Störungen Ihrer Benutzer zu vermeiden. Weitere Informationen finden Sie unter Migrieren in den Standard-Berechtigungsmodus im Amazon-QLDB-Entwicklerhandbuch.

Anforderungssyntax

```
PATCH /ledgers/name/permissions-mode HTTP/1.1
Content-type: application/json

{
   "PermissionsMode": "string"
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Ledgers.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

PermissionsMode

Der Berechtigungsmodus, der dem Ledger zugewiesen werden soll. Folgende Parameterwerte sind möglich:

- `ALLOW_ALL`: Ein Legacy-Berechtigungsmodus, der die Zugriffskontrolle mit Granularität auf API-Ebene für Ledgers ermöglicht.

  Dieser Modus ermöglicht Benutzern, die `SendCommand`-API-Berechtigung für dieses Ledger zum Ausführen aller PartiQL-Befehle (daher `ALLOW_ALL`) auf beliebigen Tabellen im angegebenen Ledger auszuführen. Dieser Modus ignoriert alle IAM-Berechtigungs-Richtlinien auf Tabellen- oder Befehls-Ebene, die Sie für das Ledger erstellen.
- `STANDARD`: (Empfohlen) Ein Berechtigungsmodus, der Zugriffskontrolle mit feinerer Granularität für Ledger, Tabellen und PartiQL-Befehle ermöglicht.

  Standardmäßig verweigert dieser Modus alle Benutzer-Anforderungen zur Ausführung von PartiQL-Befehlen für Tabellen in diesem Ledger. Damit PartiQL-Befehle ausgeführt werden können, müssen Sie IAM-Berechtigungs-Richtlinien für bestimmte Tabellen-Ressourcen und PartiQL-Aktionen erstellen, zusätzlich zu den `SendCommand` API-Berechtigung für den Ledger. Weitere Informationen finden Sie unter Erste Schritte mit dem Standard-Berechtigungsmodus im Amazon-QLDB-Entwicklerhandbuch.

> **Note**
>
> Wir empfehlen dringend, den `STANDARD` Berechtigungsmodus, um die Sicherheit Ihrer Ledger-Daten zu maximieren.

Typ: Zeichenfolge

Zulässige Werte: `ALLOW_ALL | STANDARD`

Erforderlich: Ja

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
```

```
    "Arn": "string",
    "Name": "string",
    "PermissionsMode": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

Arn

Der Amazon-Ressourcenname (ARN) für den Ledger.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

Name

Der Name des Ledgers.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

Der aktuelle Berechtigungsmodus des Ledgers.

Typ: Zeichenfolge

Zulässige Werte: `ALLOW_ALL | STANDARD`

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidParameterException

Ein oder mehrere Parameter in der Anforderung sind ungültig.

HTTP Status Code: 400

ResourceNotFoundException

Die angegebene Ressource ist nicht vorhanden.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS -Befehlszeilenschnittstelle
- AWS SDK für .NET
- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für JavaScript V3
- AWS SDK für PHP V3
- AWS SDK für Python
- AWS SDK für Ruby V3

Amazon QLDB-Sitzung

Die folgenden Aktionen werden von Amazon QLDB-Sitzung unterstützt:

- SendCommand

SendCommand

Service: Amazon QLDB Session

Sendet einen Befehl an ein Amazon QLDB-Ledger.

> **Note**
>
> Anstatt direkt mit dieser API zu interagieren, empfehlen wir, den QLDB-Treiber oder die QLDB-Shell zu verwenden, um Datentransaktionen in einem Ledger auszuführen.
>
> - Wenn Sie mit einem AWS SDK arbeiten, verwenden Sie den QLDB-Treiber. Der Treiber bietet eine abstrakte Ebene über dieser QLDB-Sitzungsdaten-API und verwaltet den `SendCommand` Vorgang für Sie. Informationen und eine Liste der unterstützten Programmiersprachen finden Sie unter Erste Schritte mit dem Treiber im Amazon QLDB Developer Guide.
> - Wenn Sie mit AWS Command Line Interface (AWS CLI) arbeiten, verwenden Sie die QLDB-Shell. Die Shell ist eine Befehlszeilenschnittstelle, die den QLDB-Treiber verwendet, um mit einem Ledger zu interagieren. Weitere Informationen finden Sie unter Zugreifen auf Amazon QLDB mithilfe der QLDB-Shell.

Anforderungssyntax

```
{
   "AbortTransaction": {
   },
   "CommitTransaction": {
      "CommitDigest": blob,
      "TransactionId": "string"
   },
   "EndSession": {
   },
   "ExecuteStatement": {
      "Parameters": [
         {
            "IonBinary": blob,
            "IonText": "string"
         }
      ],
      "Statement": "string",
```

```
      "TransactionId": "string"
   },
   "FetchPage": {
      "NextPageToken": "string",
      "TransactionId": "string"
   },
   "SessionToken": "string",
   "StartSession": {
      "LedgerName": "string"
   },
   "StartTransaction": {
   }
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

AbortTransaction

Befehl zum Abbrechen der aktuellen Transaktion.

Typ: AbortTransactionRequest Objekt

Erforderlich: Nein

CommitTransaction

Befehl zum Festschreiben der angegebenen Transaktion.

Typ: CommitTransactionRequest Objekt

Erforderlich: Nein

EndSession

Befehl zum Beenden der aktuellen Sitzung.

Typ: EndSessionRequest Objekt

Erforderlich: Nein

ExecuteStatement

Befehl zum Ausführen einer Anweisung in der angegebenen Transaktion.

Typ: ExecuteStatementRequest Objekt

Erforderlich: Nein

FetchPage

Befehl zum Abrufen einer Seite.

Typ: FetchPageRequest Objekt

Erforderlich: Nein

SessionToken

Gibt das Sitzungstoken für den aktuellen Befehl an. Ein Sitzungstoken ist während der gesamten Dauer der Sitzung konstant.

Führen Sie den `StartSession` Befehl aus, um ein Sitzungstoken zu erhalten. Dies `SessionToken` ist für jeden nachfolgenden Befehl erforderlich, der während der aktuellen Sitzung ausgegeben wird.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 4. Maximale Länge beträgt 1024 Zeichen.

Pattern: `^[A-Za-z-0-9+/=]+$`

Erforderlich: Nein

StartSession

Befehl zum Starten einer neuen Sitzung. Ein Sitzungstoken wird als Teil der Antwort abgerufen.

Typ: StartSessionRequest Objekt

Erforderlich: Nein

StartTransaction

Befehl zum Starten einer neuen Transaktion.

Typ: StartTransactionRequest Objekt

Erforderlich: Nein

Antwortsyntax

```
{
   "AbortTransaction": {
      "TimingInformation": {
         "ProcessingTimeMilliseconds": number
      }
   },
   "CommitTransaction": {
      "CommitDigest": blob,
      "ConsumedIOs": {
         "ReadIOs": number,
         "WriteIOs": number
      },
      "TimingInformation": {
         "ProcessingTimeMilliseconds": number
      },
      "TransactionId": "string"
   },
   "EndSession": {
      "TimingInformation": {
         "ProcessingTimeMilliseconds": number
      }
   },
   "ExecuteStatement": {
      "ConsumedIOs": {
         "ReadIOs": number,
         "WriteIOs": number
      },
      "FirstPage": {
         "NextPageToken": "string",
         "Values": [
            {
               "IonBinary": blob,
               "IonText": "string"
            }
         ]
      },
      "TimingInformation": {
         "ProcessingTimeMilliseconds": number
      }
   },
   "FetchPage": {
      "ConsumedIOs": {
```

```
            "ReadIOs": number,
            "WriteIOs": number
         },
         "Page": {
            "NextPageToken": "string",
            "Values": [
               {
                  "IonBinary": blob,
                  "IonText": "string"
               }
            ]
         },
         "TimingInformation": {
            "ProcessingTimeMilliseconds": number
         }
      },
      "StartSession": {
         "SessionToken": "string",
         "TimingInformation": {
            "ProcessingTimeMilliseconds": number
         }
      },
      "StartTransaction": {
         "TimingInformation": {
            "ProcessingTimeMilliseconds": number
         },
         "TransactionId": "string"
      }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

AbortTransaction

Enthält die Details der abgebrochenen Transaktion.

Typ: AbortTransactionResult Objekt

CommitTransaction

Enthält die Details der festgeschriebenen Transaktion.

Typ: CommitTransactionResult Objekt

EndSession

Enthält die Details der beendeten Sitzung.

Typ: EndSessionResult Objekt

ExecuteStatement

Enthält die Details der ausgeführten Anweisung.

Typ: ExecuteStatementResult Objekt

FetchPage

Enthält die Details der abgerufenen Seite.

Typ: FetchPageResult Objekt

StartSession

Enthält die Details der gestarteten Sitzung, die ein Sitzungstoken enthält. Dies `SessionToken` ist für jeden nachfolgenden Befehl erforderlich, der während der aktuellen Sitzung ausgegeben wird.

Typ: StartSessionResult Objekt

StartTransaction

Enthält die Details der gestarteten Transaktion.

Typ: StartTransactionResult Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

BadRequestException

Wird zurückgegeben, wenn die Anfrage falsch formatiert ist oder einen Fehler enthält, z. B. einen ungültigen Parameterwert oder einen fehlenden erforderlichen Parameter.

HTTP Status Code: 400

CapacityExceededException

Wird zurückgegeben, wenn die Anfrage die Verarbeitungskapazität des Ledgers überschreitet.

HTTP Status Code: 400

InvalidSessionException

Wird zurückgegeben, wenn die Sitzung nicht mehr existiert, weil das Zeitlimit überschritten wurde oder sie abgelaufen ist.

HTTP Status Code: 400

LimitExceededException

Wird zurückgegeben, wenn ein Ressourcenlimit, z. B. die Anzahl der aktiven Sitzungen, überschritten wird.

HTTP Status Code: 400

OccConflictException

Wird zurückgegeben, wenn eine Transaktion aufgrund eines Fehlers in der Überprüfungsphase von Optimistic Concurrency Control (OCC) nicht in das Journal geschrieben werden kann.

HTTP Status Code: 400

RateExceededException

Wird zurückgegeben, wenn die Anzahl der Anfragen den zulässigen Durchsatz überschreitet.

HTTP Status Code: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-Befehlszeilenschnittstelle
- AWS-SDK für .NET
- AWS-SDK für C++
- AWS-SDK für Go
- AWS-SDK für Java V2

- AWSSDK für JavaScript V3
- AWS-SDK für PHP V3
- AWS-SDK für Python
- AWS-SDK für Ruby V3

Datentypen

Die folgenden Datentypen werden von Amazon QLDB unterstützt:

- JournalKinesisStreamDescription
- JournalS3ExportDescription
- KinesisConfiguration
- LedgerEncryptionDescription
- LedgerSummary
- S3EncryptionConfiguration
- S3ExportConfiguration
- ValueHolder

Die folgenden Datentypen werden von Amazon QLDB Session unterstützt:

- AbortTransactionRequest
- AbortTransactionResult
- CommitTransactionRequest
- CommitTransactionResult
- EndSessionRequest
- EndSessionResult
- ExecuteStatementRequest
- ExecuteStatementResult
- FetchPageRequest
- FetchPageResult
- IOUsage
- Page

- StartSessionRequest
- StartSessionResult
- StartTransactionRequest
- StartTransactionResult
- TimingInformation
- ValueHolder

Amazon QLDB

The following data types are supported by Amazon QLDB:

- JournalKinesisStreamDescription
- JournalS3ExportDescription
- KinesisConfiguration
- LedgerEncryptionDescription
- LedgerSummary
- S3EncryptionConfiguration
- S3ExportConfiguration
- ValueHolder

JournalKinesisStreamDescription

Bedienung: Amazon QLDB

Informationen über einen Amazon QLDB-Journal-Stream, einschließlich des Amazon-Ressourcennamens (ARN), des Stream-Namens, der Erstellungszeit, des aktuellen Status und der Parameter der ursprünglichen Stream-Erstellungsanfrage.

Inhalt

KinesisConfiguration

Die Konfigurationseinstellungen des Amazon Kinesis Data Streams Streams-Ziels für einen QLDB-Journalstream.

Typ: KinesisConfiguration Objekt

Erforderlich: Ja

LedgerName

Der Name des Ledgers.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

RoleArn

Der Amazon-Ressourcenname (ARN) der IAM-Rolle, die QLDB-Berechtigungen für einen Journalstream zum Schreiben von Datensätzen in eine Kinesis Data Streams-Ressource gewährt.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

Erforderlich: Ja

Status

Der aktuelle Status des QLDB-Journaldatenstroms.

Typ: Zeichenfolge

Zulässige Werte: `ACTIVE | COMPLETED | CANCELED | FAILED | IMPAIRED`

Erforderlich: Ja

StreamId

Die UUID (in Base62-codiertem Text dargestellt) des QLDB-Journaldatenstroms.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Erforderlich: Ja

StreamName

Der benutzerdefinierte Name des QLDB-Journaldatenstroms.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

Arn

Der Amazon-Ressourcenname (ARN) des QLDB-Journaldatenstroms.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

Required: No

CreationTime

Datum und Uhrzeit im Epochenzeitformat, an dem der QLDB-Journalstream erstellt wurde. (Das Epochenzeitformat ist die Anzahl der Sekunden, die seit 00:00:00 Uhr am 1. Januar 1970 UTC vergangen sind.)

Typ: Zeitstempel

Required: No

ErrorCause

Die Fehlermeldung, die den Grund beschreibt, warum ein Stream den Status `IMPAIRED` oder hat`FAILED`. Dies gilt nicht für Streams mit anderen Statuswerten.

Typ: Zeichenfolge

Zulässige Werte: `KINESIS_STREAM_NOT_FOUND | IAM_PERMISSION_REVOKED`

Required: No

ExclusiveEndTime

Das exklusive Datum und die Uhrzeit, die angeben, wann der Stream endet. Wenn dieser Parameter nicht definiert ist, wird der Stream auf unbestimmte Zeit ausgeführt, bis Sie ihn abbrechen.

Typ: Zeitstempel

Required: No

InclusiveStartTime

Datums- und Uhrzeitangaben zum Zeitpunkt, ab dem das Streamen von Journaldaten gestartet werden soll.

Typ: Zeitstempel

Required: No

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK für Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

JournalS3ExportDescription

Bedienung: Amazon QLDB

Informationen zu einem Journalexportauftrag, einschließlich des Buchnamens, der Export-ID, der Erstellungszeit, des aktuellen Status und der Parameter der ursprünglichen Exporterstellungsanforderung.

Inhalt

ExclusiveEndTime

Das exklusive Enddatum und die Uhrzeit für den Bereich der Journalinhalte, die in der ursprünglichen Exportanforderung angegeben wurden.

Typ: Zeitstempel

Erforderlich: Ja

ExportCreationTime

Datum und Uhrzeit im Epoch- und Zeitformat der Erstellung des Exportauftrags. (Das Epochenzeitformat ist die Anzahl der Sekunden, die seit 00:00:00 Uhr am 1. Januar 1970 UTC vergangen sind.)

Typ: Zeitstempel

Erforderlich: Ja

ExportId

Die UUID (dargestellt in Base62-kodiertem Text) des Journalexportjobs.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Erforderlich: Ja

InclusiveStartTime

Das inklusive Startdatum und die Startzeit für den Bereich der Journalinhalte, die in der ursprünglichen Exportanforderung angegeben wurden.

Typ: Zeitstempel

Erforderlich: Ja

LedgerName

Der Name des Ledgers.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

RoleArn

Der Amazon-Ressourcenname (ARN) der IAM-Rolle, die QLDB-Berechtigungen für einen Journalexportauftrag für Folgendes gewährt:

- Schreiben Sie Objekte in Ihren Amazon Simple Storage Service (Amazon S3) -Bucket.
- (Optional) Verwenden Sie Ihren vom Kunden verwalteten Schlüssel in AWS Key Management Service (AWS KMS) für die serverseitige Verschlüsselung Ihrer exportierten Daten.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

Erforderlich: Ja

S3ExportConfiguration

Der Amazon Simple Storage Service (Amazon S3) -Bucket-Speicherort, in den ein Journalexportjob den Journalinhalt schreibt.

Typ: S3ExportConfiguration Objekt

Erforderlich: Ja

Status

Der aktuelle Status des Journalexportjobs.

Typ: Zeichenfolge

Zulässige Werte: `IN_PROGRESS | COMPLETED | CANCELLED`

Erforderlich: Ja

OutputFormat

Das Ausgabeformat der exportierten Journaldaten.

Typ: Zeichenfolge

Zulässige Werte: `ION_BINARY | ION_TEXT | JSON`

Required: No

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK für Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

KinesisConfiguration

Bedienung: Amazon QLDB

Die Konfigurationseinstellungen des Amazon Kinesis Data Streams-Ziels für einen Amazon QLDB-Journalstream.

Inhalt

StreamArn

Der Amazon-Ressourcenname (ARN) der Kinesis Data Streams-Ressource.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

Erforderlich: Ja

AggregationEnabled

Ermöglicht QLDB, mehrere Datensätze in einem einzelnen Kinesis-Data Streams-Datensatz zu veröffentlichen, wodurch die Anzahl der pro API-Aufruf gesendeten Datensätze erhöht wird.

Standard: `True`

> **Important**
>
> Die Datensatzaggregation hat wichtige Auswirkungen auf die Verarbeitung von Datensätzen und erfordert eine Disaggregation in Ihrem Stream-Konsumenten. Weitere Informationen finden Sie unter KPL – Die wichtigsten Konzepte und Datenproduzent – Disaggregation im Amazon Kinesis Data Streams-Entwicklerhandbuch.

Typ: Boolesch

Required: No

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK für Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

LedgerEncryptionDescription

Service: Amazon QLDB

Informationen zur Verschlüsselung von Daten im Ruhezustand in einem Amazon-QLDB-Ledger. Dazu gehören der aktuelle Status, der Schlüssel in AWS Key Management Service (AWS KMS) und der Zeitpunkt, an dem der Schlüssel nicht mehr zugänglich war (im Falle eines Fehlers).

Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand im Amazon-QLDB-Entwicklerhandbuch.

Inhalt

EncryptionStatus

Der aktuelle Status der Verschlüsselung im Ruhezustand für den Ledger. Dabei kann es sich um einen der folgenden Werte handeln:

- `ENABLED`: Die Verschlüsselung ist mit dem angegebenen Schlüssel vollständig aktiviert.
- `UPDATING`: Der Ledger verarbeitet aktiv die angegebene Schlüsseländerung.

  Die wichtigsten Änderungen in QLDB sind asynchron. Der Ledger ist vollständig zugänglich, ohne dass sich dies auf die Leistung auswirkt, während die Schlüsseländerung verarbeitet wird. Die Zeit, die für die Aktualisierung eines Schlüssels benötigt wird, variiert je nach Ledger-Größe.
- `KMS_KEY_INACCESSIBLE`: Der angegebene kundenverwaltete KMS-Schlüssel ist nicht zugänglich und der Ledger ist beeinträchtigt. Entweder wurde der Schlüssel deaktiviert oder gelöscht oder die Erteilungen für den Schlüssel wurden widerrufen. Wenn ein Ledger beeinträchtigt ist, ist er nicht zugänglich und akzeptiert keine Lese- oder Schreibanforderungen.

  Ein beeinträchtigtes Ledger kehrt automatisch in einen aktiven Zustand zurück, nachdem Sie die Erteilungen für den Schlüssel wiederhergestellt oder den deaktivierten Schlüssel erneut aktiviert haben. Das Löschen eines kundenverwalteten KMS-Schlüssels ist jedoch irreversibel. Nachdem ein Schlüssel gelöscht wurde, können Sie nicht mehr auf die Ledger zugreifen, die mit diesem Schlüssel geschützt sind, und die Daten können nicht dauerhaft wiederhergestellt werden.

Typ: Zeichenfolge

Zulässige Werte: `ENABLED | UPDATING | KMS_KEY_INACCESSIBLE`

Erforderlich: Ja

KmsKeyArn

Der Amazon-Ressourcenname (ARN) des vom Kunden verwalteten KMS-Schlüssels, den der Ledger für die Verschlüsselung im Ruhezustand verwendet. Wenn dieser Parameter nicht definiert ist, verwendet der Ledger einen - AWS eigenen KMS-Schlüssel für die Verschlüsselung. Es wird angezeigt`AWS_OWNED_KMS_KEY`, wenn die Verschlüsselungskonfiguration des Ledgers auf den AWS -eigenen KMS-Schlüssel aktualisiert wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

Erforderlich: Ja

InaccessibleKmsKeyDateTime

Das Datum und die Uhrzeit im Epochenzeitformat, zu der der AWS KMS Schlüssel im Falle eines Fehlers zum ersten Mal unzugänglich wurde. (Epochisches Zeitformat ist die Anzahl der Sekunden, die seit dem 12:00:00 Uhr 1. Januar 1970 UTC verstrichen sind.)

Dieser Parameter ist nicht definiert, wenn auf den AWS KMS Schlüssel zugegriffen werden kann.

Typ: Zeitstempel

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie unter:

- AWS SDK für C++
- AWS SDK für Go
- AWS SDK für Java V2
- AWS SDK für Ruby V3

LedgerSummary

Bedienung: Amazon QLDB

Informationen zu einem Ledger, einschließlich seines Namens, Status und Zeitpunkt seiner Erstellung.

Inhalt

CreationDateTime

Datum und Uhrzeit im Epoch- und Uhrzeit im Epoch- und Uhrzeit im Epoch- und Uhrzeit. (Das Epochenzeitformat ist die Anzahl der Sekunden, die seit 00:00:00 Uhr am 1. Januar 1970 UTC vergangen sind.)

Typ: Zeitstempel

Required: No

Name

Der Name des Ledgers.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Nein

State

Der aktuelle Status der Ledger.

Typ: Zeichenfolge

Zulässige Werte: `CREATING | ACTIVE | DELETING | DELETED`

Required: No

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK für Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

S3EncryptionConfiguration

Bedienung: Amazon QLDB

Die Verschlüsselungseinstellungen, die von einem Amazon Simple Storage Service (Amazon S3) - Bucket.

Inhalt

ObjectEncryptionType

Der Amazon S3 S3-Objektverschlüsselungstyp.

Weitere Informationen zu serverseitigen Verschlüsselungsoptionen in Amazon S3 finden Sie unter Schutz von Daten mithilfe serverseitiger Verschlüsselung im Amazon S3 S3-Entwicklerhandbuch.

Typ: Zeichenfolge

Zulässige Werte: `SSE_KMS | SSE_S3 | NO_ENCRYPTION`

Erforderlich: Ja

KmsKeyArn

Der Amazon-Ressourcenname (ARN) eines symmetrischen Verschlüsselungsschlüssels in AWS Key Management Service (AWS KMS). Hinweis: Amazon S3 unterstützt keine asymmetrischen KMS-Schlüssel.

Sie müssen eine angeben`KmsKeyArn`, wenn Sie `SSE_KMS` als die angeben`ObjectEncryptionType`.

`KmsKeyArn`ist nicht erforderlich, wenn Sie `SSE_S3` als die angeben`ObjectEncryptionType`.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 1600 Zeichen.

Required: No

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK für Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

S3ExportConfiguration

Bedienung:Amazon QLDB

Der Name des Amazon Simple Storage Service (Amazon S3) -Bucket, in dem ein Journal-Exportauftrag den Journalinhalt ablegt.

Inhalt

Bucket

Der Name des Amazon S3 S3-Bucket, in dem ein Journal-Exportjob den Journalinhalt schreibt.

Der Name des Bucket muss den Amazon S3 S3-Bucket-Benennungskonventionen entsprechen. Weitere Informationen finden Sie unter Bucket-Einschränkungen und -Limits im Amazon S3 S3-Entwicklerhandbuch.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Höchstlänge = 255 Zeichen.

Pattern: `^[A-Za-z-0-9-_.]+$`

Erforderlich: Ja

EncryptionConfiguration

Die Verschlüsselungseinstellungen, die von einem Journal-Exportauftrag verwendet werden, um Daten in einen Amazon S3 S3-Bucket zu schreiben.

Typ: S3EncryptionConfiguration Objekt

Erforderlich: Ja

Prefix

Das Präfix für den Amazon S3 S3-Bucket, in dem ein Journal-Exportjob den Journalinhalt schreibt.

Das Präfix muss den Regeln und Einschränkungen für die Benennung von Amazon S3 S3-Schlüsseln entsprechen. Weitere Informationen finden Sie unter Objektschlüssel und Metadaten im Amazon S3 S3-Entwicklerhandbuch.

Im Folgenden finden Sie Beispiele für gültige`Prefix` Werte:

- `JournalExports-ForMyLedger/Testing/`
- `JournalExports`
- `My:Tests/`

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Maximale Länge beträgt 128 Zeichen.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

ValueHolder

Bedienung:Amazon QLDB

Eine Struktur, die einen Wert in mehreren Kodierungsformaten enthalten kann.

Inhalt

IonText

Ein Amazon Ion-Klartextwert, der in einer`ValueHolder` Struktur enthalten ist.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 1048576 Zeichen.

Required: No

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

Amazon QLDB-Sitzung

Die folgenden Datentypen werden von Amazon QLDB-Sitzung unterstützt:

- AbortTransactionRequest
- AbortTransactionResult
- CommitTransactionRequest
- CommitTransactionResult
- EndSessionRequest
- EndSessionResult
- ExecuteStatementRequest

- ExecuteStatementResult
- FetchPageRequest
- FetchPageResult
- IOUsage
- Page
- StartSessionRequest
- StartSessionResult
- StartTransactionRequest
- StartTransactionResult
- TimingInformation
- ValueHolder

AbortTransactionRequest

Bedienung: Amazon QLDB Session

Enthält die Details der Transaktion, die abgebrochen werden soll.

Inhalt

Die Mitglieder dieser Ausnahmestruktur sind kontextabhängig.

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK für Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

AbortTransactionResult

Bedienung:Amazon QLDB Session

Enthält die Details der abgebrochenen Transaktion.

Inhalt

TimingInformation

Enthält serverseitige Leistungsinformationen für den Befehl.

Typ: TimingInformation Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

CommitTransactionRequest

Bedienung:Amazon QLDB Session

Enthält die Details der Transaktion, die bestätigt werden soll.

Inhalt

CommitDigest

Gibt den Commit-Digest für die zu übernehmende Transaktion an. Für jede aktive Transaktion muss der Commit-Digest übergeben werden. QLDB validiert den Commit`CommitDigest` und weist ihn mit einem Fehler zurück, wenn der auf dem Client berechnete Digest nicht mit dem von QLDB berechneten Digest übereinstimmt.

Der Zweck des`CommitDigest` Parameters besteht darin, sicherzustellen, dass QLDB eine Transaktion genau dann durchführt, wenn der Server genau die vom Client gesendeten Anweisungen verarbeitet hat, in derselben Reihenfolge, in der der Client sie gesendet hat, und ohne Duplikate.

Typ: Base64-kodiertes Binärdatenobjekt

Erforderlich: Ja

TransactionId

Gibt die Transaktions-ID der Transaktion an, für die übergeben werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go

- AWS-SDK für Java V2
- AWS SDK für Ruby V3

CommitTransactionResult

Bedienung:Amazon QLDB Session

Enthält die Details der bestätigten Transaktion.

Inhalt

CommitDigest

Der Commit-Digest der Commit-Transaktion.

Typ: Base64-kodiertes Binärdatenobjekt

Required: No

ConsumedIOs

Enthält Metriken über die Anzahl der E/A-Anfragen, die verbraucht wurden.

Typ: IOUsage Objekt

Required: No

TimingInformation

Enthält serverseitige Leistungsinformationen für den Befehl.

Typ: TimingInformation Objekt

Required: No

TransactionId

Die Transaktions-ID der bestätigten Transaktion.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

EndSessionRequest

Bedienung: Amazon QLDB Session

Gibt eine Anfrage zum Beenden der Sitzung an.

Inhalt

Die Mitglieder dieser Ausnahmestruktur sind kontextabhängig.

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK für Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

EndSessionResult

Bedienung:Amazon QLDB Session

Enthält die Details der beendeten Sitzung.

Inhalt

TimingInformation

Enthält serverseitige Leistungsinformationen für den Befehl.

Typ: TimingInformation Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

ExecuteStatementRequest

Bedienung: Amazon QLDB Session

Gibt eine Anforderung zur Ausführung einer Anweisung an.

Inhalt

Statement

Gibt die Anweisung der Anfrage an.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Höchstlänge = 1 000 Zeichen.

Erforderlich: Ja

TransactionId

Gibt die Transaktions-ID der Anforderung an.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Erforderlich: Ja

Parameters

Gibt die Parameter für die parametrisierte Anforderung in der Anforderung an.

Typ: Array von ValueHolder-Objekten

Required: No

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK für Go

- AWS-SDK für Java V2
- AWS SDK für Ruby V3

ExecuteStatementResult

Bedienung:Amazon QLDB Session

Enthält die Details der ausgeführten Anweisung.

Inhalt

ConsumedIOs

Enthält Metriken über die Anzahl der E/A-Anfragen, die verbraucht wurden.

Typ: IOUsage Objekt

Required: No

FirstPage

Enthält die Details der ersten abgerufenen Seite.

Typ: Page Objekt

Required: No

TimingInformation

Enthält serverseitige Leistungsinformationen für den Befehl.

Typ: TimingInformation Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

FetchPageRequest

Bedienung:Amazon QLDB Session

Gibt die Details der abzurufenden Seite an.

Inhalt

NextPageToken

Gibt das nächste Seiten-Token der abzurufenden Seite an.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 4. Maximale Länge von 1 024.

Pattern: `^[A-Za-z-0-9+/=]+$`

Erforderlich: Ja

TransactionId

Gibt die Transaktions-ID der abzurufenden Seite an.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

FetchPageResult

Bedienung:Amazon QLDB Session

Enthält die Seite, die abgerufen wurde.

Inhalt

ConsumedIOs

Enthält Metriken über die Anzahl der E/A-Anfragen, die verbraucht wurden.

Typ: IOUsage Objekt

Required: No

Page

Enthält Details der abgerufenen Seite.

Typ: Page Objekt

Required: No

TimingInformation

Enthält serverseitige Leistungsinformationen für den Befehl.

Typ: TimingInformation Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

IOUsage

Bedienung:Amazon QLDB Session

Enthält I/O-Nutzungsmetriken für einen Befehl, der aufgerufen wurde.

Inhalt

ReadIOs

Die Anzahl der Lese-I/O-Anforderungen, die der Befehl gestellt hat.

Type: Long

Required: No

WriteIOs

Die Anzahl der Schreib-I/O-Anforderungen, die der Befehl gestellt hat.

Type: Long

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

Page

Bedienung:Amazon QLDB Session

Enthält Details der abgerufenen Seite.

Inhalt

NextPageToken

Das Token der nächsten Seite.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 4. Maximale Länge von 1 024.

Pattern: `^[A-Za-z-0-9+/=]+$`

Required: No

Values

Eine Struktur, die Werte in mehreren Kodierungsformaten enthält.

Typ: Array von ValueHolder-Objekten

Required: No

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

StartSessionRequest

Bedienung:Amazon QLDB Session

Legt eine Anforderung fest, eine neue Sitzung zu starten.

Inhalt

LedgerName

Der Name des Ledgers, für das eine neue Sitzung gestartet werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 32 Zeichen.

Pattern: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

StartSessionResult

Bedienung:Amazon QLDB Session

Enthält die Details der gestarteten Sitzung.

Inhalt

SessionToken

Sitzungstoken der gestarteten Sitzung. Dies`SessionToken` ist für jeden nachfolgenden Befehl erforderlich, der während der aktuellen Sitzung ausgegeben wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 4. Maximale Länge von 1 024.

Pattern: `^[A-Za-z-0-9+/=]+$`

Required: No

TimingInformation

Enthält serverseitige Leistungsinformationen für den Befehl.

Typ: TimingInformation Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

StartTransactionRequest

Bedienung: Amazon QLDB Session

Spezifiziert eine Anfrage zum Starten einer Transaktion.

Inhalt

Die Mitglieder dieser Ausnahmestruktur sind kontextabhängig.

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK für Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

StartTransactionResult

Bedienung:Amazon QLDB Session

Enthält die Details der gestarteten Transaktion.

Inhalt

TimingInformation

Enthält serverseitige Leistungsinformationen für den Befehl.

Typ: TimingInformation Objekt

Required: No

TransactionId

Die Transaktions-ID der gestarteten Transaktion.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 22.

Pattern: `^[A-Za-z-0-9]+$`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

TimingInformation

Bedienung:Amazon QLDB Session

Enthält serverseitige Leistungsinformationen für einen Befehl. Amazon QLDB erfasst Zeitinformationen zwischen dem Empfang der Anfrage und dem Senden der entsprechenden Antwort.

Inhalt

ProcessingTimeMilliseconds

Die Zeit, die QLDB für die Verarbeitung des Befehls aufgewendet hat, gemessen in Millisekunden.

Type: Long

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

ValueHolder

Bedienung:Amazon QLDB Session

Eine Struktur, die einen Wert in mehreren Kodierungsformaten enthalten kann.

Inhalt

IonBinary

Ein Amazon Ion-Binärwert, der in einer`ValueHolder` Struktur enthalten ist.

Typ: Base64-kodiertes Binärdatenobjekt

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge von 131072.

Required: No

IonText

Ein Amazon Ion-Klartextwert, der in einer`ValueHolder` Struktur enthalten ist.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge von 1048576.

Required: No

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS-SDKs finden Sie unter:

- AWS-SDK für C++
- AWS-SDK for Go
- AWS-SDK für Java V2
- AWS SDK für Ruby V3

Häufige Fehler

In diesem Abschnitt sind Fehler aufgeführt, die häufig bei den API-Aktionen aller AWS-Services auftreten. Informationen zu Fehlern, die spezifisch für eine API-Aktion für diesen Service sind, finden Sie unter dem Thema für diese API-Aktion.

AccessDeniedException

Sie haben keinen ausreichenden Zugriff zum Durchführen dieser Aktion.

HTTP Status Code: 400

IncompleteSignature

Die Anforderungssignatur entspricht nicht den AWS-Standards.

HTTP Status Code: 400

InternalFailure

Die Anforderungsverarbeitung ist fehlgeschlagen, da ein unbekannter Fehler, eine Ausnahme oder ein Fehler aufgetreten ist.

HTTP Status Code: 500

InvalidAction

Die angeforderte Aktion oder Operation ist ungültig. Überprüfen Sie, ob die Aktion ordnungsgemäß eingegeben wurde.

HTTP Status Code: 400

InvalidClientTokenId

Das angegebene X.509-Zertifikat oder die AWS-Zugriffsschlüssel-ID ist nicht in unseren Datensätzen vorhanden.

HTTP Status Code: 403

NotAuthorized

Sie haben keine Berechtigung zum Ausführen dieser Aktion.

HTTP Status Code: 400

OptInRequired

Die AWS-Zugriffsschlüssel-ID benötigt ein Abonnement für den Service.

HTTP Status Code: 403

RequestExpired

Die Anforderung hat den Service mehr als 15 Minuten nach dem Datumsstempel oder mehr als 15 Minuten nach dem Ablaufdatum der Anforderung erreicht (z. B. für vorsignierte URLs) oder der Datumsstempel auf der Anforderung liegt mehr als 15 Minuten in der Zukunft.

HTTP Status Code: 400

ServiceUnavailable

Die Anforderung ist aufgrund eines temporären Fehlers des Servers fehlgeschlagen.

HTTP Status Code: 503

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTP Status Code: 400

ValidationError

Die Eingabe erfüllt nicht die von einem AWS-Service definierten Einschränkungen.

HTTP Status Code: 400

Geläufige Parameter

Die folgende Liste enthält die Parameter, die alle Aktionen zum Signieren von Signature-Version-4-Anforderungen mit einer Abfragezeichenfolge verwenden. Alle aktionsspezifischen Parameter werden im Thema für diese Aktion aufgelistet. Weitere Informationen zu Signature Version 4 finden Sie unter Signieren vonAWS API-Anfragen im IAM-Benutzerhandbuch.

Action

Die auszuführende Aktion.

Typ: Zeichenfolge

Erforderlich: Ja

Version

Die API-Version, für die die Anforderung geschrieben wurde, ausgedrückt im Format JJJJ-MM-TT.

Typ: Zeichenfolge

Erforderlich: Ja

X-Amz-Algorithm

Der Hashalgorithmus, den Sie zum Erstellen der Anforderungssignatur verwendet haben.

Bedingung: Geben Sie diesen Parameter an, wenn Sie Authentifizierungsinformationen in eine Abfragezeichenfolge anstatt in den HTTP-Autorisierungsheader aufnehmen.

Typ: Zeichenfolge

Zulässige Werte: `AWS4-HMAC-SHA256`

Required: Conditional

X-Amz-Credential

Der Wert des Anmeldeinformationsumfangs. Dabei handelt es sich um eine Zeichenfolge, die Ihren Zugriffsschlüssel, das Datum, die gewünschte Region und eine Zeichenfolge zur Beendigung („aws4_request“) beinhaltet. Der Wert wird im folgenden Format ausgedrückt: Zugriffsschlüssel/JJJJMMTT/Region/Service/aws4_request.

Weitere Informationen finden Sie unter Erstellen einer signiertenAWS API-Anfrage im IAM-Benutzerhandbuch.

Bedingung: Geben Sie diesen Parameter an, wenn Sie Authentifizierungsinformationen in eine Abfragezeichenfolge anstatt in den HTTP-Autorisierungsheader aufnehmen.

Typ: Zeichenfolge

Required: Conditional

X-Amz-Date

Das Datum, das zum Erstellen der Signatur verwendet wird. Das Format muss das ISO 8601-Basisformat (JJJJMMTT'T'SSMMSS'Z') sein. Die folgende Datumszeit ist beispielsweise ein gültiger X-Amz-Date-Wert: `20120325T120000Z`.

Bedingung: X-Amz-Date ist bei allen Anforderungen optional. Damit kann das Datum überschrieben werden, das zum Signieren von Anforderungen verwendet wird. Wenn der Date-Header im ISO 8601-Basisformat angegeben ist, ist X-Amz-Date nicht erforderlich. Wenn X-Amz-

Date verwendet wird, überschreibt es immer den Wert des Date-Headers. Weitere Informationen finden Sie unter Elemente einerAWS API-Anforderungssignatur im IAM-Benutzerhandbuch.

Typ: Zeichenfolge

Required: Conditional

X-Amz-Security-Token

Das temporäre Sicherheitstoken, das durch einen Anruf vonAWS Security Token Service (AWS STS) abgerufen wurde. Eine Liste der Services, die temporäre Sicherheitsanmeldeinformationen von unterstütztenAWS STS AWS-Services, finden Sie unter, die mit IAM arbeiten im IAM-Benutzerhandbuch.

Bedingung: Wenn Sie temporäre Sicherheitsanmeldeinformationen von nutzenAWS STS, müssen Sie das Sicherheitstoken einschließen.

Typ: Zeichenfolge

Required: Conditional

X-Amz-Signature

Gibt die hex-codierte Signatur an, die aus der zu signierenden Zeichenfolge und dem abgeleiteten Signaturschlüssel berechnet wurde.

Bedingung: Geben Sie diesen Parameter an, wenn Sie Authentifizierungsinformationen in eine Abfragezeichenfolge anstatt in den HTTP-Autorisierungsheader aufnehmen.

Typ: Zeichenfolge

Required: Conditional

X-Amz-SignedHeaders

Gibt alle HTTP-Header an, die als Teil der kanonischen Anforderung enthalten waren. Weitere Informationen zur Angabe signierter Header finden Sie unter Erstellen einer signiertenAWS API-Anfrage im IAM-Benutzerhandbuch.

Bedingung: Geben Sie diesen Parameter an, wenn Sie Authentifizierungsinformationen in eine Abfragezeichenfolge anstatt in den HTTP-Autorisierungsheader aufnehmen.

Typ: Zeichenfolge

Required: Conditional

Kontingente und Limits in Amazon QLDB

In diesem Abschnitt werden die aktuellen Kontingente — auch als Grenzwerte bezeichnet — in Amazon QLDB beschrieben.

Themen

Standardkontingente

QLDB hat die folgenden Standardkontingente, die auch unter Amazon QLDB-Endpoints und Kontingente in der aufgeführt sind Allgemeine AWS-Referenz. Diese Kontingente geltenAWS-Konto pro Region. Um eine Kontingenterhöhung für Ihr Konto in einer Region anzufordern, können Sie die Service-Quotas-Konsole verwenden.

Melden Sie sich bei anAWS Management Console und öffnen Sie die Service-Quotas-Konsole unter https://console.aws.amazon.com/servicequotas/.

Ressource	Standardkontingent
Die maximale Anzahl aktiver Ledger, die Sie in diesem Konto in der aktuellen Region erstellen können	5
Die maximale Anzahl aktiver Journalexporte nach Amazon S3 pro Ledger	2
Die maximale Anzahl von aktiven Journal-Streams zu Kinesis Data Streams pro Ledger	5

Feste Kontingente

Zusätzlich zu den Standardkontingenten hat QLDB die folgenden festen Kontingente pro Ledger. Diese Kontingente können nicht durch die Verwendung von Service Quotas erhöht werden:

Ressource	Feste Kontingente
Anzahl gleichzeitig aktiver Sitzungen	1500
Anzahl der aktiven Tabellen	20
Gesamtanzahl der Tabellen (aktiv und inaktiv) **Note** In QLDB gelten gelöschte Tabellen als inaktiv und werden auf dieses Gesamtkontingent angerechnet.	40
Anzahl der Indizes pro Tabelle	5
Anzahl der Dokumente in einer Transaktion	40
Anzahl der Revisionen, die in einer Transaktion redigiert werden sollen	1
Dokumentgröße (im `IonBinary` Format kodiert)	128 KB
Größe des Ausweisungsparameters (`IonBinary` Format)	128 KB
Größe des Ausweisungsparameters (`IonText`Format)	1 MB
Länge der Anweisungszeichenfolge	100 000 Zeichen
Größe der Transaktion	4 MB

Ressource	Feste Kontingente
Transaktions-Timeout	30 Sekunden
Ablaufzeitraum für abgeschlossene Journalexportaufträge	7 Tage
Ablaufzeitraum für Beendigungs-Journal-Streams	7 Tage

Ledger-Kontingente

Um eine Erhöhung des Ledger-Quotas für Ihr Konto in einer Region anzufordern, können Sie die Service-Quotas-Konsole verwenden.

Öffnen Sie die Service-Quotas-Konsole unter https://console.aws.amazon.com/servicequotas/.

In einigen QLDB-Anwendungsfällen ist je nach Geschäftswachstum eine zunehmende Anzahl von LedgernAWS-Konto pro Region erforderlich. Beispielsweise könnte es sein, dass Sie spezielle Ledger erstellen müssen, um Kunden oder Daten zu isolieren. In diesem Fall sollten Sie erwägen, eine Architektur mit mehreren Konten zu nutzen, um mit QLDB-Kontingenten zu arbeiten. Weitere Informationen finden Sie unter Account Silo Isolation imAWS Whitepaper SaaS Tenant Isolation Strategies.

Dokumentengröße

Die maximale Größe für ein Dokument, das in dem`IonBinary` Format codiert ist, beträgt 128 KB. Wir können keine genaue Grenze für die Größe eines Dokuments im`IonText` Format angeben, da die Konvertierung von Text in Binärdatei je nach Struktur der einzelnen Dokumente erheblich variiert. QLDB unterstützt Dokumente mit offenem Inhalt, sodass jede einzigartige Dokumentstruktur die Größenberechnung ändert.

Transaktionsgröße

Die maximale Größe für eine Transaktion in QLDB ist 4 MB. Die Größe einer Transaktion ergibt sich aus der Summe der folgenden Faktoren.

Deltas

Das Dokumentänderungen, die von allen Anweisungen innerhalb der Transaktion generiert werden. Bei einer Transaktion, die sich auf mehrere Dokumente auswirkt, ist die gesamte Delta-Größe die Summe der einzelnen Deltas jedes betroffenen Dokuments.

Metadaten

Die systemgenerierten Transaktionsmetadaten, die jedem betroffenen Dokument zugeordnet sind.

Indizes

Wenn ein Index für eine Tabelle definiert ist, die von der Transaktion betroffen ist, generiert der Indexeintrag auch einen Delta.

Verlauf

Da alle Dokumentrevisionen in QLDB beibehalten werden, werden alle Transaktionen auch an die Historie angehängt.

Einfügungen — Für jedes in eine Tabelle eingefügte Dokument wird auch eine Kopie in die Verlaufstabelle eingefügt. Beispiel: Ein neu eingefügtes 100-KB Dokument generiert mindestens 200 KB von Deltas in einer Transaktion. (Dies ist eine grobe Schätzung, die keine Metadaten oder Indizes enthält.)

Aktualisierungen — Jede Aktualisierung eines Dokuments, auch für ein einzelnes Feld, erstellt eine neue Revision des gesamten Dokuments in der Historie, plus oder minus des Deltas der Aktualisierung. Das bedeutet, dass ein kleines Update in einem großen Dokument weiterhin ein großes Transaktions-Delta generiert. Beispiel: Durch das Hinzufügen von 2 KB an Daten in ein vorhandenes 100-KB-Dokument wird eine neue 102-KB-Revision im Verlauf erstellt. Dies beläuft sich auf mindestens 104 KB der gesamten Deltas in einer Transaktion. (Auch hier beinhaltet diese Schätzung keine Metadaten oder Indizes.)

Löschungen — Ähnlich wie bei Aktualisierungen erstellt jede Löschtransaktion eine neue Dokumentversion in der Historie. Die neu erstellte `DELETE`-Revision ist jedoch kleiner als das ursprüngliche Dokument, da sie keine Benutzerdaten hat und nur Metadaten enthält.

Benennungseinschränkungen:

In der folgenden Tabelle werden die Benennungseinschränkungen in Amazon QLDB beschrieben.

Ledger-Name Name des Journaldatenstreams	• Sie dürfen nur 1—32 alphanumerische Zeichen oder Bindestriche enthalten. • Muss einen Buchstaben oder eine Zahl für das erste und letzte Zeichen enthalten. • Darf nicht nur aus Zahlen bestehen. • Er darf keine zwei aufeinanderfolgenden Bindestriche enthalten. • Die Groß- und Kleinschreibung wird berücksichtigt.
Tabellenname	• Sie dürfen nur 1—128 alphanumerische Zeichen oder Unterstriche enthalten. • Muss einen Groß- oder Kleinbuchstaben enthalten. • Kann eine beliebige Kombination aus alphanumerischen Zeichen und Unterstrichen für die verbleibenden Zeichen enthalten. • Die Groß- und Kleinschreibung wird berücksichtigt. • Darf kein reserviertes QLDB PartiQL-Wort sein.

Verwandte zu Amazon QLDB

Die folgenden verwandten Ressourcen bieten Ihnen nützliche Informationen für die Arbeit mit diesem Service.

Themen

Technische Dokumentation

- Amazon QLDB FAQs — Häufig gestellte Fragen zum Produkt.
- Amazon QLDB-Preise —AWS Preisinformationen und Beispiele.
- AWS re:Post—AWS Community-Forum für Fragen und Antworten (Q&A).
- Amazon Ion — Entwicklerhandbücher, Benutzerhandbücher und Referenzen für das Amazon Ion-Datenformat.
- PartiQL — Ein Spezifikationsdokument und allgemeine Tutorials für die PartiQL-Abfragesprache.
- QLDB-Workshops — Workshops, die praktische Beispiele für die Verwendung von Amazon QLDB zum Erstellen von system-of-record Anwendungen bieten, einschließlich der folgenden Labs:
  - Erlernen der QLDB-Grundlagen
  - Arbeiten mit Amazon Ion und Konvertierung von Ion in und von JSON (Java)
  - Verwendung vonAWS Glue und Amazon Athena zur Aktivierung von QLDB-Daten für einen Data Lake
  - Streaming von QLDB-DB-DBInstance
- Manipulationssichere Qualitätsdaten mithilfe von Amazon QLDB — Eine AWSLösungsimplementierung, die zeigt, wie Angreifer daran gehindert werden können, hochwertige Daten zu manipulieren, indem QLDB verwendet wird, um einen genauen Verlauf der Datenänderungen zu führen. AWS Lösungsimplementierungen helfen Ihnen dabei, häufig auftretende Probleme zu lösen und die Nutzung zu beschleunigenAWS.

GitHub Repositorien

Treiber

- .NET-Treiber — Eine .NET-Implementierung des QLDB-Treibers.
- Go-Treiber — Eine Go-Implementierung des QLDB-Treibers.
- Java-Treiber — Eine Java-Implementierung des QLDB-Treibers.
- Treiber Node.js — Eine Node.js -Implementierung des QLDB-Treibers.
- Python-Treiber — Eine Python-Implementierung des QLDB-Treibers.

Befehlszeilen-Shell

- QLDB-Shell — Eine Python- und Rust-Implementierung einer Befehlszeilenschnittstelle für die QLDB-Transaktionsdaten-API.

Beispielanwendungen

- Java-DMV-Anwendung — Eine Tutorial-Anwendung, die auf einem Anwendungsfall des Department of Motor Vehicles (DMV) basiert. Es demonstriert grundlegende Operationen und bewährte Methoden für die Verwendung von QLDB und des QLDB-Treibers für Java.
- .NET-DMV-Anwendung — Eine DMV-basierte Tutorial-Anwendung, die grundlegende Operationen und bewährte Methoden für die Verwendung von QLDB und des QLDB-Treibers für .NET demonstriert.
- DMV-Anwendung Node.js — Eine DMV-basierte Tutorial-Anwendung, die grundlegende Operationen und bewährte Methoden für die Verwendung von QLDB und des QLDB-Treibers für Node.js demonstriert.
- Python-DMV-Anwendung — Eine DMV-basierte Tutorial-Anwendung, die grundlegende Operationen und bewährte Methoden für die Verwendung von QLDB und des QLDB-Treibers für Python demonstriert.
- Ledger Loader — Ein Java-Framework für das asynchrone Laden von Daten in ein QLDB-Ledger mit hoher Geschwindigkeit unter Verwendung eines unterstützten Lieferkanals (AWS DMSAmazon SQS, Amazon SNS, Kinesis Data Streams, Amazon MSK oder EventBridge).
- Exportprozessor — Ein erweiterbares Java-Framework, das die Verarbeitung von QLDB-Exporten in Amazon S3 übernimmt, indem es die Ausgabe des Exports liest und die exportierten Blöcke nacheinander durchläuft.

- QLDB-Streams sind ein Beispiel für Lambda in Python — Eine Anwendung, die demonstriert, wie QLDB-Streams mithilfe einerAWS Lambda Funktion zum Senden von QLDB-Daten an ein Amazon SNS SNS-Thema verwendet werden, für das eine Amazon SQS SQS-Warteschlange abonniert ist.
- Beispiel für die OpenSearch Integration von QLDB-Streams — Eine Python-Anwendung, die demonstriert, wie Amazon OpenSearch Service mithilfe von Streams in QLDB integriert wird.
- Anwendung mit doppelter Eingabe — Eine Java-Anwendung, die demonstriert, wie eine Finanzbuchanwendung mit doppelter Eingabe mithilfe von QLDB modelliert wird.
- QLDB KVS for Node.js — Eine einfache Schnittstellenbibliothek zum Speichern von Schlüsselwerten für QLDB mit zusätzlichen Funktionen zur Dokumentenüberprüfung.

Amazon Ion und PartiQL

- Amazon Ion-Bibliotheken — Vom Ion-Team unterstützte Bibliotheken, Tools und Dokumentation.
- PartiQL-Implementierung — Die Implementierung, Spezifikation und Tutorials für PartiQL.

AWSBlogbeiträge und Artikel

- Wie Earnin seinen Ledger-Service mithilfe von Amazon QLDB aufgebaut hat (16. Februar 2023) — Beschreibt, wie Earnin QLDB verwendet hat, um einen Ledger-Service aufzubauen, der seinen Benutzern eine moderne und voll funktionsfähige mobile Finanzanwendung bietet.
- Exportieren und analysieren Sie Amazon QLDB-Journaldaten mithilfe vonAWS Glue und Amazon Athena (19. Dezember 2022) — Erläutert, wie Sie die Exportfunktion in QLDB mitAWS Glue und Athena verwenden können, um Ihren buchgestützten Architekturen Berichts- und Analysefunktionen zur Verfügung zu stellen.
- Wie Shinsegae International mit Amazon QLDB das Kundenerlebnis verbessert und Fälschungen verhindert (3. August 2022) — Beschreibt, wie Shinsegae International mithilfe von Amazon QLDB einen digitalen Authentizitätsüberprüfungsservice aufgebaut hat, um Kunden über die Echtheit von Luxusgütern zu informieren und die Kauf- und Vertriebshistorie von Produkten bereitzustellen.
- BungKuSit nutzt Amazon QLDB und die ISV-Technologie von VeriDoc Global, um das Kunden- und Lieferantenerlebnis zu verbessern (29. April 2022) — Zeigt, wie ein Logistikunternehmen, BungKuSit, QLDB nutzte, um eine zentrale, transparente Plattform für die branchenübergreifende Kommunikation mit einem unveränderlichen, kryptografisch überprüfbaren Transaktionsprotokoll zu implementieren.

- Amazon QLDB als unveränderlichen Schlüsselwertspeicher mit einer REST-API und JSON verwenden (14. Februar 2022) — Stellt eine Möglichkeit vor, mit QLDB als unveränderlichen Schlüsselwertspeicher zu arbeiten und seine Auditfunktionen über eine REST-API zu nutzen.
- Aufbau eines Kernbankensystems mit Amazon QLDB (21. Januar 2022) — Zeigt, wie QLDB verwendet wird, um ein Kernbankbuchsystem aufzubauen. Es zeigt auch, warum QLDB eine gute fit-for-purpose Datenbank ist, die den Bedürfnissen der Finanzdienstleistungsbranche entspricht.
- Wie Specright Amazon QLDB verwendet, um ein rückverfolgbares Lieferkettennetzwerk aufzubauen (21. Januar 2022) — Zeigt, wie Specright QLDB verwendet, um ein rückverfolgbares Lieferkettennetzwerk zu schaffen, das es Großhandelsmarken, Einzelhändlern und Herstellern ermöglicht, wichtige Lieferkettendaten und Verpackungsspezifikationen auszutauschen.
- Wie FeMR mit Amazon QLDB kryptografisch sichere und überprüfbare medizinische Daten bereitstellt (23. Dezember 2021) — Erfahren Sie, wie Team FeMR QLDB und andereAWS verwaltete Dienste nutzte, um seine Hilfsmaßnahmen zu ermöglichen.
- Amazon QLDB-Abfragezugriffsmuster überwachen (8. November 2021) — Zeigt, wie QLDB-Transaktionen und die PartiQL-Anweisungen, die in den Transaktionen ausgeführt wurden, mit den über Amazon API Gateway empfangenen API-Anfragen verknüpft werden.
- Erstellen Sie einen einfachen CRUD-Vorgang und einen Datenstream auf Amazon QLDB mithilfe vonAWS Lambda (28. September 2021) — Zeigt, wie CRUD-Operationen (Erstellen, Lesen, Aktualisieren und Löschen) auf QLDB mithilfe vonAWS Lambda Funktionen ausgeführt werden.
- Kryptografische Überprüfung in der realen Welt mit Amazon QLDB (26. August 2021) — Erläutert den Wert der kryptografischen Überprüfung in einem QLDB-Ledger im Kontext eines realistischen Anwendungsfalls.
- Überprüfen Sie die Lieferbedingungen mit dem Accord-Projekt und Amazon QLDB: Teil 1, Teil 2 (28. Juni 2021) — Erläutert, wie die Technologie intelligenter Rechtsverträge zur Überprüfung der Lieferbedingungen mit dem Open-Source-Accord-Projekt und QLDB angewendet werden kann.
- Amazon QLDB-Datenstreaming überAWS CDK (7. Juni 2021) — Zeigt, wie Sie die verwenden,AWS Cloud Development Kit (AWS CDK) um QLDB einzurichten, die QLDB-Daten mithilfe vonAWS Lambda Funktionen aufzufüllen und QLDB-Streaming einzurichten, um die Datenstabilität der Ledger-Daten zu gewährleisten.
- Demo zur fein abgestuften Zugriffskontrolle in QLDB (1. Juni 2021) — Zeigt, wie Sie mit detaillliertenAWS Identity and Access Management (IAM) -Berechtigungen für ein QLDB-Ledger beginnen.

- Stream-Verarbeitung mit DynamoDB Streams und QLDB-Streams (23. November 2020) — Bietet einen kurzen Vergleich zwischen DynamoDB-Streams und QLDB-Streams sowie Tipps für den Einstieg in diese.
- Streaming-Daten von Amazon QLDB bis OpenSearch (19. August 2020) — Beschreibt, wie Daten von QLDB an Amazon OpenSearch Service gestreamt werden, um die Rich-Textsuche und Downstream-Analysen wie Aggregation oder Metriken zwischen Datensätzen zu unterstützen.
- Wie ich Daten mit NodeJS nahezu in Echtzeit von Amazon QLDB nach DynamoDB gestreamt habe (7. Juli 2020) — Beschreibt, wie Daten von QLDB nach DynamoDB gestreamt werden, um einstellige Latenzzeiten und unendlich skalierbare Schlüsselwertabfragen zu unterstützen.
- Aufbau einer GraphQL-Schnittstelle zu Amazon QLDB mitAWS AppSync: Teil 1, Teil 2 (4. Mai 2020) — Erläutert, wie QLDB integriert undAWS AppSync eine vielseitige, GraphQL-gestützte API auf der Grundlage eines QLDB-Ledgers bereitgestellt wird.

Medien

AWS-Videos

- AWSTutorials und Demos: QLDB to Aurora Streaming (17. März 2023; 21 Minuten) — Dieses Video erklärt die grundlegenden Konzepte zur Implementierung der QLDB-Streaming-Funktion und zeigt, wie Datenstreaming von QLDB zu einer Amazon Aurora MySQL-Downstream-DB eingerichtet wird.
- ArcBlock: Nutzung von Amazon QLDB zum Aufbau einer dezentralen Identitätslösung (31. Mai 2022; 4 Minuten) — ArcBlock führt durch die dezentrale Identitätslösung, die sie mithilfe von QLDB entwickelt haben.
- AWSre:Invent 2020: Amazon QLDB als system-of-trust Datenbank für zentrale Geschäftsanwendungen verwenden (5. Februar 2021; 30 Minuten) — Erfahren Sie, wie die ersten Amazon QLDB-Benutzer die einzigartigen Eigenschaften der Ledger-Datenbank für Datenherkunft und kryptografische Überprüfbarkeit genutzt haben, um Aufzeichnungssysteme mit integrierter Datenintegrität zu implementieren.
- AWSre:Invent 2020: Aufbau einer serverlosen Anwendung mit Amazon QLDB (5. Februar 2021; 28 Minuten) — Erfahren Sie, wie Sie eine voll funktionsfähige serverlose Anwendung erstellen, testen und optimierenAWS Lambda, indem Sie Amazon QLDB mit Diensten wie Amazon Kinesis und Amazon DynamoDB kombinieren.
- AWSre:Invent 2020: Entwicklung von auditbasierten Apps zur Wahrung der Datenintegrität mit Amazon QLDB (5. Februar 2021; 18 Minuten) — Diese Sitzung befasst sich mit den Problemen,

die Amazon QLDB lösen kann, beantwortet Ihre Fragen dazu, wann und warum Sie eine Ledger-Datenbank verwenden sollten, und zeigt Anwendungsfälle von Kunden wie Osano.

- So speichern Sie unveränderliche Protokolle mithilfe von Amazon QLDB mit.NET-Anwendungen zentral (7. Dezember 2020; 10 Minuten) — Eine Demonstration, wie Sie QLDB mit .NET-Anwendungen verwenden, um unveränderliche Protokolle zentral zu speichern.
- Virtueller Workshop: Aufbau von Ledger-Based Systems of Record with QLDB and Java —AWS Online Tech Talks (29. Juli 2020; 87 Minuten) — Ein von Dozenten geführter Workshop, der durch das Working With Ion Immersion Day Lab führt, um ein Dataloader-Programm unter Verwendung der Amazon Ion-Bibliothek und des QLDB-Treibers für Java zu erstellen.
- Entwickler-Workshop:AWS Using's Immutable Ledger Database QLDB auf ABT Node (22. Juni 2020; 34 Minuten) — Eine step-by-step Anleitung zur Einrichtung und Konfiguration von QLDB auf ABT Node. Es erklärt auch, wie Sie den Blockchain Explorer und ArcBlock die Boarding Gate Blocklets installieren und konfigurieren, um eine Verbindung zu QLDB herzustellen.
- AWSTech Talk: Die Perspektive eines Kunden zum Aufbau einer ereignisgesteuerten System-of-Record-Anwendung mit Amazon QLDB (31. März 2020; 29 Minuten) — Ein technischer Vortrag von Matt Lewis —AWS Data Hero und Chief Architect der Driver and Vehicle Licensing Agency (DVLA) in Großbritannien —, in dem der Anwendungsfall der DVLA für QLDB und die ereignisgesteuerte Architektur ihrer Anwendung erläutert wird.
- AWSre:Invent 2019: Warum Sie eine Ledger-Datenbank benötigen: BMW, DVLA und Sage besprechen Anwendungsfälle (5. Dezember 2019; 47 Minuten) — Eine Präsentation der Kunden BMW, der britischen Regierungsorganisation DVLA und Sage, in der die Gründe für die Verwendung einer Ledger-Datenbank erörtert und ihre Anwendungsfälle für QLDB vorgestellt werden.
- AWSre:Invent 2019: Amazon QLDB: Ein Techniker erklärt, warum dies ein entscheidender Faktor ist (5. Dezember 2019; 50 Minuten) — Eine Präsentation von Andrew Certain (AWSDistinguished Engineer), in der die einzigartige Architektur von QLDB, an erster Stelle steht, zusammen mit ihren verschiedenen Innovationen erörtert wird. Es umfasst kryptografisches Hashing, Merkle-Bäume, Replikation mehrerer Availability Zones und PartiQL-Unterstützung.
- Building Applications with Amazon QLDB, a Unity Ledger Database —AWS Online Tech Talks (19. November 2019; 51 Minuten) — Ein ausführlicher Tech-Vortrag, der die einzigartigen Funktionen von QLDB und Einzelheiten zur Verwendung der Kernfunktionen beschreibt. Dazu gehören die Abfrage des vollständigen Verlaufs Ihrer Daten, die kryptografische Überprüfung von Dokumenten und das Entwerfen eines Datenmodells.

Podcasts

- Warum entscheiden sich Kunden für Amazon QLDB? (5. Juli 2020; 33 Minuten) — Eine Diskussion, in der die Definition einer Ledger-Datenbank erklärt wird, wie sie sich von einer Blockchain unterscheidet und wie Kunden sie heute verwenden.

Allgemeine AWS-Ressourcen

- Kurse und Workshops – Links zu rollenbasierten und speziellen Kursen sowie Übungen im Selbststudium zur Verbesserung Ihrer AWS-Kompetenzen und Erweiterung Ihrer praktischen Erfahrung.
- AWS-Entwicklerzentrum – Entdecken Sie Tutorials, laden Sie Tools herunter und erfahren Sie mehr über Veranstaltungen für AWS-Entwickler.
- AWS-Entwickler-Tools – Links zu Entwickler-Tools, SDKs, IDE-Toolkits und Befehlszeilen-Tools für die Entwicklung und Verwaltung von AWS-Anwendungen.
- Ressourcenzentrum für die ersten Schritte – Erfahren Sie, wie Sie Ihr AWS-Konto einrichten, der AWS-Community beitreten und Ihre erste Anwendung starten.
- Praktische step-by-step Tutorials — Schritt-Anleitungen zum Starten Ihrer ersten Anwendung aufAWS.
- AWS Whitepaper – Links zu einer umfangreichen Liste technischer AWS-Whitepaper zu Themen wie Architektur, Sicherheit und Wirtschaftlichkeit. Diese Whitepaper wurden von AWS-Lösungsarchitekten und anderen technischen Experten verfasst.
- AWS Support-Center – Hub für die Erstellung und Verwaltung Ihrer AWS Support-Fälle. Stellt darüber hinaus Links zu weiteren nützlichen Ressourcen bereit, beispielsweise Foren, häufig gestellten technischen Fragen, Status der Service-Integrität und AWS Trusted Advisor.
- AWS Support— Primäre Website für Informationen zuAWS Support one-on-one, einem reaktionsschnellen Support-Channel, der Sie bei der Erstellung und Ausführung von Anwendungen in der Cloud unterstützt.
- Kontakt – Zentraler Kontaktpunkt für Fragen zu AWS-Abrechnung, Konten, Ereignissen Missbrauch und anderen Problemen.
- Nutzungsbedingungen für die AWS-Website – Detaillierte Informationen zu unseren Copyright- und Markenbestimmungen, Ihrem Konto, den Lizenzen und anderen Themen.

Versionsverlauf für Amazon QLDB B-QLDB B-QLDB

Die folgende Tabelle beschreibt wichtige Änderungen in jeder Version von Amazon QLDB sowie die entsprechenden Updates im Amazon QLDB-Entwicklerhandbuch für Amazon QLDB-Entwicklerhandbuch. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie den RSS-Feed abonnieren.

- API-Version: 2019-01-02
- Letzte Aktualisierung der Dokumentation: 3. Januar 2023

Änderung	Beschreibung	Datum
Aktualisierte IAM-Leitlinien	Aktualisierter Leitfaden, angepasst an die bewährten IAM-Methoden. Weitere Informationen finden Sie unter Bewährte IAM-Methoden.	3. Januar 2023
Aktualisierung derAWS verwalteten Richtlinien	Amazon QLDB hat die bestehendenAWS verwaltet en Richtlinien aktualisi ert`AmazonQLDBFullAcce ss` und`AmazonQLD BConsoleFullAccess` . Diese Richtlinien verfügen über eine neue Berechtigung, die es Prinzipalen ermöglicht, Dokumentrevisionen mithilfe einer gespeicherten PartiQL-P rozedur zu redigieren. Weitere Informationen finden Sie unter AWSVerwaltete Richtlinien für Amazon QLDB.	04. November 2022
Redigierung von Daten	Amazon QLDB unterstützt jetzt die gespeicherte`REDACT_RE`	03. November 2022

	`VISION` PartiQL-Prozedur für Ledger, die am oder nach dem 22. Juli 2021 erstellt wurden. Mithilfe dieser gespeicherten Prozedur können Sie inaktive Dokumentversionen in der Historie dauerhaft löschen und trotzdem die allgemeine Datenintegrität Ihres Hauptbuchs beibehalten. Weitere Informationen finden Sie unter Schwärzung von Dokumentversionen.	
Node.js Treiber v3	Der Amazon QLDB-Treiber für Node.js Version 3.0 ist jetzt allgemein verfügbar. Diese Version führt die Unterstützung für dieAWS SDK for JavaScript v3 ein. Versionshinweise finden Sie im GitHub Repository awslabs/amazon-qldb-driver-nodejs.	26. September 2022
Go-Treiber v3	Der Amazon QLDB-Treiber für Go Version 3.0 ist jetzt allgemein verfügbar. Diese Version führt die Unterstützung für dieAWS SDK for Go v2 ein. Versionshinweise finden Sie im GitHub Repository awslabs/amazon-qldb-driver-go.	11. August 2022

Neuer PartiQL-Abfrageeditor	Ein neuer PartiQL-Abfrageeditor in der Amazon QLDB-Konsole ist jetzt allgemein verfügbar. Der neue QLDB PartiQL-Editor bietet eine verbesserte Oberfläche für das Erstellen von Abfragen, das Debuggen von Transaktionen und das Durchsuchen von Ergebnissen. Informationen zum Öffnen und Verwenden des Editors finden Sie unter Zugreifen auf Amazon QLDB mithilfe der Konsole.	22. Juni 2022
Ion Object Mapper für .NET-Treiber	Der Amazon QLDB-Treiber für .NET Version 1.3 führt die Unterstützung für den Ion Object Mapper ein. Mit dieser Funktion können Sie die manuelle Konvertierung zwischen Amazon Ion-Typen und nativen C#-Typen vollständig umgehen. Die vollständige Änderungshistorie des Ion-Objektmappers finden Sie in der Datei CHANGELOG.md im GitHub Repositor `yamzn/ion-object-mapper-dotnet` .	19. Januar 2022

Exportformat für JSON-Journale	Amazon QLDB unterstützt jetzt das JSON Lines-Ausgabeformat für Journalexporte. Weitere Informationen finden Sie unter Exportieren von Journaldaten aus Amazon QLDB.	21. Dezember 2021
Fehlerbehebung bei Amazon QLDB	Es wurde ein neues Thema zur Problembehandlung hinzugefügt, das Anleitungen für eine zusammenfassende Liste der häufigsten Fehler enthält, die bei der Verwendung von Amazon QLDB auftreten können.	8. Dezember 2021
Start einer neuen Region	Amazon QLDB B-B ist jetzt in der Region Kanada (Zentral) verfügbar. Eine vollständige Liste der verfügbaren Regionen finden Sie unter Amazon QLDB-Endpunkte und Kontingente in der Allgemeine Amazon Web Services-Referenz.	11. November 2021

Dienstübergreifende Confused-Deputy-Prävention	Amazon QLDB unterstützt jetzt die Verwendung von Kontext-Schlüsseln `aws:SourceArn` und `aws:SourceAccount` globale Bedingungskontexts chlüssel in den IAM-Resso urcenrichtlinien, um das Confused-Deputy-Problem zu umgehen. Weitere Informati onen finden Sie unter Vermeidung des verwirrten Stellvertreters im dienstübe rgreifenden Szenario.	8. November 2021
Amazon QLDB-Shell v2	Version 2.0 der Amazon QLDB-Shell, die in Rust geschrieben ist, ist jetzt allgemein verfügbar. Versionshinweise finden Sie im GitHub Repository awslabs/ amazon-qldb-shell.	14. Oktober 2021

Aktualisierung derAWS verwalteten Richtlinien	Amazon QLDB hat die bestehendenAWS verwaltet en Richtlinien aktualisi ert`AmazonQLDBFullAcce ss` und`AmazonQLD BConsoleFullAccess` . Diese Richtlinien haben neu hinzugefügte Berechtig ungen, die es Schulleitern ermöglichen, jede IAM-Rolle nressource in Ihrem Konto an den QLDB-Service zu übergeben. Dies ist für alle Journalexport- und Stream-Anforderungen erforderlich. Weitere Informationen finden Sie unter AWSVerwaltete Richtlinien für Amazon QLDB.	2. September 2021
Vom Kunden verwalteteAWS KMS Schlüssel	Amazon QLDB unterstüt zt jetzt Verschlüsselung im Ruhezustand mithilfe von kundenverwalteten Schlüssel n inAWS Key Management Service (AWS KMS) für neue Ledger-Ressourcen. Weitere Informationen finden Sie unter Verschlüsselung im Ruhezusta nd in Amazon QLDB.	22. Juli 2021

Aktualisierung derAWS verwalteten Richtlinie	Amazon QLDB hat die bestehendeAWS verwaltete Richtlinie aktualisiert `AmazonQLDBReadOnly`, um eine doppelte `qldb:GetBlock` Aktion zu entfernen, die zuvor zweimal aufgeführt war. Weitere Informationen finden Sie unter AWSVerwaltete Richtlinien für Amazon QLDB.	1. Juli 2021
Start einer neuen Region	Amazon QLDB B-B ist jetzt in der Region Europa (London) verfügbar. Eine vollständige Liste der verfügbaren Regionen finden Sie unter Amazon QLDB-Endpunkte und Kontingente in der Allgemeine Amazon Web Services-Referenz.	24. Juni 2021

Aktualisierung derAWS verwalteten Richtlinien	Amazon QLDB hat die bestehendenAWS verwaltet en Richtlinien aktualisi ert`AmazonQLDBFullAcce ss` und`AmazonQLD BConsoleFullAccess` . Diese Richtlinien haben neu hinzugefügte Berechtigungen, die es Prinzipalen ermöglich en, den Berechtigungsmodus in allen Ledgern zu aktualisi eren und alle PartiQL-Befehle in allen`STANDARD` Berechtig ungsbüchern auszuführen. Weitere Informationen finden Sie unter AWSVerwaltete Richtlinien für Amazon QLDB.	27. Mai 2021
Modus „Standardberechtig ungen“	Amazon QLDB unterstützt jetzt einen`STANDARD` Berechtig ungsmodus für Ledger-Re ssourcen. Mit dem Standardb erechtigungsmodus können Sie den Zugriff mit feinerer Granularität für Ledger, Tabellen und PartiQL-Befehle steuern. Weitere Informationen finden Sie unter Erste Schritte mit dem Standardberechtigu ngsmodus im Standardb erechtigungsmodus.	27. Mai 2021

PartiQL-Statement-Statistiken	Die Funktion für PartiQL-Statement-Statistiken ist jetzt im Abfrage-Editor der Amazon QLDB-Konsole verfügbar. Weitere Informationen finden Sie unter Abrufen von Statistiken zu laufenden Anweisungen.	24. Mai 2021
Tutorial: Daten mit einemAWS SDK verifizieren	Es wurde ein step-by-step Tutorial mit Codebeispielen hinzugefügt, das zeigt, wie ein Revisions-Hash und ein Block-Hash in Amazon QLDB mithilfe der QLDB-API über einAWS SDK überprüft werden. Weitere Informationen finden Sie unter Tutorial: Daten mit einemAWS SDK überprüfen.	6. Mai 2021
.NET-Treiber v1.2	Der Amazon QLDB-Treiber für .NET Version 1.2 ist jetzt allgemein verfügbar. Diese Version führt asynchrone APIs ein. Versionshinweise finden Sie im GitHub Repository awslabs/amazon-qldb-driver-dotnet.	01. April 2021
PartiQL`DROP INDEX` QL-Anweisung	PartiQL in Amazon QLDB unterstützt jetzt die DROP INDEX-Anweisung. Weitere Informationen finden Sie unter Löschen von Indizes.	3. März 2021

PartiQL-Statement-Statistiken	Die Funktion für PartiQL-Statement-Statistiken ist jetzt in der neuesten Version des Amazon QLDB-Treibers für alle unterstützten Sprachen verfügbar, einschließlich .NET, Go und Python. Weitere Informationen finden Sie unter Abrufen von Statistiken zu laufenden Anweisungen.	25. Februar 2021
TypeScript Schnellstart-Tutorial	Im Schnellstart-Tutorial für den Amazon QLDB-Treiber für Node.js wurden TypeScript Codebeispiele hinzugefügt.	28. Dezember 2020
PartiQL-Statement-Statistiken	Die neueste Version des Amazon QLDB-Treibers für Java und Node.js bietet jetzt Statistiken zur Anweisungsausführung, die Ihnen helfen können, PartiQL-Anweisungen effizienter auszuführen. Weitere Informationen finden Sie unter Abrufen von Statistiken zu laufenden Anweisungen.	22. Dezember 2020
Hinweise zum Fahrer-Kochbuch und Schnellstart-Tutorials	Kochbuchreferenzen für die Go- und Node.js Treiber, Schnellstart-Tutorials für die Java- und Python-Treiber und eine Anleitung für die Arbeit mit Amazon Ion in QLDB wurden hinzugefügt. Weitere Informationen finden Sie unter Erste Schritte mit dem Amazon QLDB-Treiber.	24. November 2020

Amazon QLDB-Shell v1.1	Version 1.1 der Amazon QLDB-Shell ist jetzt allgemein verfügbar. Versionshinweise finden Sie im GitHub Repository awslabs/amazon-qldb-shell.	26. Oktober 2020
Amazon QLDB-Treiber für Go	Der Amazon QLDB B-Treiber für Go ist jetzt allgemein verfügbar. Dieser Treiber ist Open Source GitHub und ermöglicht es Ihnen, den zu verwendenAWS SDK for Go, um mit der Transaktionsdaten-API von QLDB zu interagieren. Versionshinweise finden Sie im GitHub Repository awslabs/amazon-qldb-driver-go.	20. Oktober 2020
Empfehlungen zur Treibereinrichtung von Node.js	Es wurde ein neuer Abschnitt hinzugefügt, der Setup-Empfehlungen für den Amazon QLDB-Treiber für Node.js enthält, einschließlich der Reduzierung der Latenz durch die Wiederverwendung von Verbindungen mit Keep-Alive.	16. Oktober 2020
Indexerstellung für nicht leere Tabellen	Amazon QLDB unterstützt jetzt die Indexerstellung für nicht leere Tabellen. Weitere Informationen finden Sie unter Verwalten von Indizes.	30. September 2020

Optimierung der Abfragelei stung	Es wurde ein neuer Abschnitt hinzugefügt, der Abfragebe schränkungen in Amazon QLDB beschreibt und Anleitungen zur Optimierung der Abfrageleistung angesicht s dieser Einschränkungen enthält.	18. September 2020
Kochbuch-Referenz für den Java-Treiber	Es wurde eine Cookbook-Referenz hinzugefügt, die Codebeispiele für gängige Anwendungsfälle für den Amazon QLDB-Treiber für Java zeigt.	3. September 2020
Sitzungsmanagement mit dem Fahrer	Es wurde ein neuer Abschnitt hinzugefügt, der einen Überblick über die Sitzungsv erwaltung mit dem Treiber in Amazon QLDB gibt und beschreibt, wie der QLDB-Treiber Sitzungen bei der Ausführung von Datentran saktionen verarbeitet.	1. September 2020
Java-Treiber v2.0	Der Amazon QLDB-Trei ber für Java Version 2.0 ist jetzt allgemein verfügbar. Versionshinweise finden Sie im GitHub Repository awslabs/ amazon-qldb-driver-java.	28. August 2020

Node.js Treiber v2.0	Der Amazon QLDB-Treiber für Node.js Version 2.0 ist jetzt allgemein verfügbar. Versionshinweise finden Sie im GitHub Repository awslabs/amazon-qldb-driver-nodejs.	27. August 2020
Ähnliche Informationen	Ein neues Thema wurde hinzugefügt, das Links für verwandte Informationen und zusätzliche Ressourcen enthält, die Ihnen helfen, Amazon QLDB zu verstehen und damit zu arbeiten.	24. August 2020
Python-Treiber v3.0	Der Amazon QLDB-Treiber für Python Version 3.0 ist jetzt allgemein verfügbar. Versionshinweise finden Sie im GitHub Repository awslabs/amazon-qldb-driver-python.	20. August 2020
Amazon QLDB-Treiber für Go	Eine Vorschauversion des Amazon QLDB-Treibers für Go ist jetzt verfügbar. Mit diesem Treiber können Sie den verwendenAWS SDK for Go, um mit der Transaktionsdaten-API von QLDB zu interagieren. Weitere Informationen finden Sie unter Amazon QLDB Treiber für Go (Vorversion).	6. August 2020

Kochbuch-Referenz für den Python-Treiber	Es wurde eine Cookbook-Referenz hinzugefügt, die Codebeispiele für gängige Anwendungsfälle für den Amazon QLDB-Treiber für Python zeigt.	24. Juli 2020
Eindeutige IDs in Amazon QLDB	Es wurde ein neuer Abschnitt hinzugefügt, der die Eigenschaften und Verwendungsrichtlinien von Unique IDs in Amazon QLDB beschreibt.	9. Juli 2020
AWS CloudFormationRessource für Journal-Streams	Amazon QLDB unterstützt jetzt eine Ressource zum Erstellen von Journal-Streams mithilfe einerAWS CloudFormation Vorlage. Weitere Informationen finden Sie in der AWS::QLDB::StreamRessource im AWS CloudFormationBenutzerhandbuch.	9. Juli 2020
Kochbuch-Referenz für den .NET-Treiber	Es wurde eine Cookbook-Referenz hinzugefügt, die Codebeispiele für gängige Anwendungsfälle für den Amazon QLDB-Treiber für.NET zeigt.	1. Juli 2020

Amazon QLDB-Treiber für.NET	Der Amazon QLDB-Treiber für.NET ist jetzt allgemein verfügbar. Dieser Treiber ist Open Source GitHub und ermöglicht es Ihnen, den zu verwendenAWS SDK for .NET, um mit der Transaktionsdaten-API von QLDB zu interagieren. Versionshinweise finden Sie im GitHub Repository awslabs/amazon-qldb-driver-dotnet.	26. Juni 2020
Amazon QLDB-Treiber für Node.js	Der Amazon QLDB-Treiber für Node.js ist jetzt allgemein verfügbar. Dieser Treiber ist Open Source GitHub und ermöglicht es Ihnen, dasAWS SDK für JavaScript in Node.js zu verwenden, um mit der Transaktionsdaten-API von QLDB zu interagieren. Versionshinweise finden Sie im amazon-qldb-driver-nodejs GitHub awslabs/-Repository.	5. Juni 2020

Amazon QLDB-Journalstreams	Mit Amazon QLDB können Sie jetzt einen Stream erstellen, der jede in Ihr Journal übernommene Dokumentenrevision erfasst und diese Daten nahezu in Echtzeit an Amazon Kinesis Data Streams übermittelt. Weitere Informationen finden Sie unter Streamen von Journaldaten aus Amazon QLDB.	19. Mai 2020
Amazon Ion-Codebeispiele	Es wurden Codebeispiele hinzugefügt, die Amazon Ion-Daten in einem Amazon QLDB-Ledger mithilfe des QLDB-Treibers für Java, Node.js und Python abfragen und verarbeiten. Weitere Informationen finden Sie unter Ion-Codebeispiele in Amazon QLDB.	12. Mai 2020
Parallelitätsmodell und Journalinhalt	Das Thema Amazon QLDB-Parallelitätsmodell wurde um Informationen zur Verwendung von Indizes zur Begrenzung von Konflikten zur optimistischen Parallelitätssteuerung (OCC) erweitert. Es wurde ein neuer Abschnitt hinzugefügt, der den Journalinhalt in Amazon QLDB beschreibt.	4. Mai 2020
Schnellstartanleitung für den Treiber Node.js	Eine Schnellstartanleitung für den Amazon QLDB-Treiber für Node.js wurde hinzugefügt.	1. Mai 2020

Amazon QLDB-Shell	Die Amazon QLDB B-Shell ist jetzt allgemein verfügbar. Diese Shell ist Open Source GitHub und bietet eine Befehlszeilenschnittstelle, mit der Sie PartiQL-Anweisunge n auf Ledger-Daten ausführen können. Versionshinweise finden Sie im amazon-ql db-shell GitHub awslabs/- Repository.	20. April 2020
Compliance-Zertifizierungen	Amazon QLDB ist jetzt fürAWS Compliance-Program me wie HIPAA und ISO zertifiziert. Weitere Informati onen finden Sie unter Konformitätsüberarbeitungen für Amazon QLDB.	3. April 2020
Erweiterte Fahreremp fehlungen	Der Abschnitt mit den Empfehlungen für Amazon QLDB-Treiber wurde erweitert, sodass er für alle unterstützten Programmiersprachen gilt.	2. April 2020
Amazon QLDB-Shell	Eine Vorschauversion der Amazon QLDB-Shell ist jetzt verfügbar. Diese Shell stellt eine Befehlszeilenschnittstelle bereit, mit der Sie PartiQL-A nweisungen für Ledger-Daten ausführen können. Weitere Informationen finden Sie unter Zugreifen auf Amazon QLDB mithilfe der QLDB-Shell (nur Daten-API) (Vorschau).	23. März 2020

Java-Treiber v1.1	Der Amazon QLDB-Treiber für Java Version 1.1 ist jetzt allgemein verfügbar. Versionshinweise finden Sie im amazon-qldb-driver-java GitHub awslabs/-Repository.	20. März 2020
Amazon QLDB-Treiber für.NET	Amazon QLDB bietet jetzt eine Vorschauversion des .NET-Treibers. Mit diesem Treiber können Sie den verwenden AWS SDK for .NET, um mit der Transaktionsdaten-API von QLDB zu interagieren. Weitere Informationen finden Sie unter Amazon QLDB-Treiber für .NET (Vorschau).	13. März 2020
Amazon QLDB-Treiber für Python	Der Amazon QLDB B-Treiber für Python ist jetzt allgemein verfügbar. Dieser Treiber ist Open Source GitHub und ermöglicht es Ihnen, den zu verwendenAWS SDK for Python (Boto3), um mit der Transaktionsdaten-API von QLDB zu interagieren. Versionshinweise finden Sie im amazon-qldb-driver-python GitHub awslabs/-Repository.	11. März 2020

UNDROP TABLEPartiQL-Anweisung und verschachtelte DML	PartiQL in Amazon QLDB unterstützt jetzt DML-Anweisungen (Data Manipulation Language), in denen verschachtelte Sammlungen als Quellen in der`FROM` Klausel angegeben werden. Weitere Informationen finden Sie in der FROM-Anweisung in der PartiQL-Referenz. QLDB PartiQL unterstützt auch die Anweisung UNDROP TABLE. Weitere Informationen finden Sie unter Löschen von Tabellen.	26. Februar 2020
Erweitertes Intro-Thema	Erweiterung der Einführung Was ist Amazon QLDB? Thema, das die neuen Abschnitte Überblick über Amazon QLDB und From relational to Ledger enthält.	24. Januar 2020
PartiQL-Referenz für unterstützte Funktionen	Die Amazon QLDB PartiQL-Referenz wurde um detaillierte Nutzungsinformationen zu den unterstützten SQL-Funktionen erweitert. Weitere Informationen finden Sie unter PartiQL-Funktionen.	2. Januar 2020

Fahrerempfehlungen und häufige Fehler	Es wurden neue Abschnitte hinzugefügt, in denen Treiberempfehlungen für den Amazon QLDB-Treiber für Java und allgemeine Fehler für alle Treibersprachen beschrieben werden.	2. Januar 2020
Start neuer Regionen	Amazon QLDB ist jetzt in den Regionen Asien-Pazifik (Singapur), Asien-Pazifik (Singapur), Asien-Pazifik (Sydney) und Europa (Frankfurt) an. Eine vollständige Liste der verfügbaren Regionen finden Sie unter Amazon QLDB-Endpunkte und Kontingente in der Allgemeine Amazon Web Services-Referenz.	19. November 2019
Amazon QLDB-Treiber für Node.js	Amazon QLDB bietet jetzt eine Vorschauversion des Treibers Node.js. Mit diesem Treiber können Sie dasAWS SDK für JavaScript in Node.js verwenden, um mit der Transaktionsdaten-API von QLDB zu interagieren. Weitere Informationen finden Sie unter Amazon QLDB-Treiber für Node.js (Vorschau).	13. November 2019

Amazon QLDB-Treiber für Python	Amazon QLDB bietet jetzt eine Vorschauversion des Python-Treibers. Mit diesem Treiber können Sie den verwenden AWS SDK for Python (Boto3), um mit der Transaktionsdaten-API von QLDB zu interagieren. Weitere Informationen finden Sie unter Amazon QLDB-Treiber für Python (Vorschau).	29. Oktober 2019
Öffentliche Veröffentlichung	Dies ist die erste veröffentlichte Version von Amazon QLDB. Diese Version enthält den Entwicklerleitfaden und die integrierte Ledger-Management-API-Referenz.	10. September 2019

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.